2

PARALLEL SYSTEMS LABORATORY: ACCESS, ALLOCATION, AND CONTROL DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

FINAL TECHNICAL REPORT

June 30, 1992

Principal Investigator: Leonard Kleinrock

Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles

PARALLEL SYSTEMS LABORATORY: ACCESS, ALLOCATION, AND CONTROL DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

FINAL TECHNICAL REPORT

June 30, 1992

Principal Investigator: Leonard Kleinrock

Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PARALLEL SYSTEMS LABORATORY: ACCESS, CONTROL, AND ALLOCATION Final Technical Report		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report July 1, 1987 - June 30, 1992
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Leonard Kleinrock		8. CONTRACT OR GRANT NUMBER(s) MDA 903-87-C-0663
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Engineering & Applied Science University of California, Los Angeles Los Angeles, CA 90024-1596		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 2496/28
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 3701 N. Fiarfax Drive Arlington, VA 22203-1714		12. REPORT DATE June 30, 1992
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PARALLEL SYSTEMS LABORATORY: ACCESS, ALLOCATION, AND CONTROL

FINAL TECHNICAL REPORT

June 30, 1992

Contract Number: MDA 903-87-C-0663
DARPA Order Number: 2496/28
Contract Period: July 1, 1987 to June 30, 1992
Report Period: July 1, 1987 to June 30, 1992

Principal Investigator: Leonard Kleinrock

(310) 825-2543

Computer Science Department
School of Engineering and Applied Science
University of California, Los Angeles

Sponsored by

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

PARALLEL SYSTEMS LABORATORY: ACCESS, ALLOCATION, AND CONTROL

Defense Advanced Research Projects Agency

Final Technical Report

June 30, 1992

This Final Technical Report covers research carried out by the Parallel Systems Laboratory: Access, Allocation and Control at UCLA under DARPA Contract Number MDA 903-87-C-0663, covering the period July 1, 1987 through June 30, 1992.

Below we restate the tasks which have been the subject of this research effort. We then give a summary of the salient results of this research in capsule form which is followed by an extensive list of publications, dissertations and theses which have been the result of this research. The contract period covered is from July 1, 1987 to June 30, 1992; however, one must note that whereas this contract formally ended on June 30, 1992, the period from July 1, 1990 to June 30, 1992 was simply a no-cost extension of funds to complete and publish results of studies that had begun during the main contract period.

The included papers summarize a number of our research activities but are only a sample of our work. Of course the more complete results are contained in the full set of publications we have listed, copies of which have been sent to the DARPA contracting office during the period of this contract.

Brief Statement of Work

The brief statement of work which describes the tasks on which we have been working during this contract are as follows:

TASK I. FUNDAMENTAL TOOLS AND ISSUES IN PARALLEL PROCESSING SYSTEMS

It is imperative that we develop evaluation tools as well as fundamental understanding of the behavior, performance and tradeoffs for parallel processing systems. We propose to develop such tools, and to evaluate the way in which the system resources interact with each other and with the demands placed upon them.

TASK II. PARALLEL PROCESSING ARCHITECTURE AND ALGORITHM DESIGN

A given set of resources can be configured to be centralized or distributed in many ways. We propose to study the effect on performance and cost of various system topologies (e.g., series, parallel, series/parallel, network), organization and architecture. We will design algorithms to operate in a parallel processing environment and will evaluate the effect of unreliable processors.

TASK III. MIGRATING AND ADAPTIVE PROCESSES

The distributed control of parallel systems has proven to be a major development in system design. We propose to study the design of distributed algorithms which adapt to the load and structure of their environment, and to evaluate their performance (delay, efficiency, complexity) in massively connected systems.

TASK IV. DISTRIBUTED DATA BASE

The coordination of a data base in a distributed environment is an extremely important issue in large systems today. We propose novel methods for accessing and updating data base systems from distributed sources using the technique of optimistic locking; this algorithm has the potential for providing higher efficiency in a variety of complex systems.

TASK V. TIGHTLY COUPLED PARALLEL PROCESSING SYSTEMS

Tightly coupled processors in a parallel processing environment may suffer serious performance degradation problems. We propose to provide tools for evaluating performance for a very wide class of parallel processing systems. We will include the effect of the number of processors available, the parallelism inherent in various graph models of computation, the arrival process of new jobs, and the randomness in the task computation times. A related issue is that of massive connectivity in massively parallel systems. We propose to evaluate the tradeoff among communications, processing and storage in these systems.

TASK VI. EXPERIMENTATION IN PARALLEL SYSTEMS

A key task in this project is to conduct a carefully staged sequence of experiments in parallel processing using the existing testbeds available in the Computer Science Department at UCLA. We have named this the Benevolent Bandit Laboratory (or BBL Project, for short). Our main purpose is to use it as a vehicle for experimenting with an operational parallel processing system; initially, we intend to use it to harness the power of underutilized workstations which, fortunately, are already connected together as a "system" through a local area network.

Brief Summary of Significant Results

During this contract period, we have published 21 papers and have had 2 additional papers accepted in professional journals. Moreover, we have contributed 2 book chapters. In addition to all of this, 6 Ph.D. students have graduated as a result of the work they have conducted under this research in addition to 6 Master's students.

Our major results fall into three somewhat overlapping areas which relate to the statement of work. These three areas are:

1. Parallel processing systems;
2. Distributed processing systems;
3. Distributed communications

In the area of parallel processing systems, we have a number of interesting results and developments. One of our major activities during this contract period was to develop what we call the Benevolent Bandit Laboratory; this is a testbed for understanding the behavior of distributed algorithms in an environment in which we recapture the idle processing capacities from a large number of workstations connected in a local area network. The concept here of course is to try to take advantage of the enormous amount of idle time experienced by most workstations. Indeed we built a shell around the workstation operating system which senses when the processor is idle for more than a minimum amount of time at which time our Benevolent Bandit shell takes over the processor and runs a background process; as soon as the user of the system needs access to his own workstation, the shell immediately relinquishes control and returns it to the user. We implemented the system, we measured it, we wrote and ran applications on it, and we analyzed the behavior of this system and of a large number of other similar systems of much greater extent both in numbers and operations. Indeed the paper by Kleinrock and Korfhage "Collecting Unused Processing Capacity: An Analysis of Transient Distributed Systems" which is included in the Appendix of this report summarizes the analytical results which we obtained for this system. The bottom line for such an operation is that one can capture the unused capacity of networked workstations and that the performance is predictable and understandable.

Another major development we achieved in our study of parallel processing systems was to understand how many processors one should use in order to effectively carry out the dependent tasks of a sequence of jobs as they flow into a parallel processing system with P processors. The problem here is if we have too many processors, the efficiency of each is low (but the response time is conveniently short); on the other hand, if we have too few processors, the efficiency of each is high (but the response time degrades). Using a combined objective function known as power, which is simply the ratio of efficiency to mean response time, we determine the optimal number of processors one should use in such a system, determine the optimum load such a system should sustain, and identify the principles of operation and insights which come out of such an analysis. The key ideas here are summarized in the paper by Kleinrock and Huang entitled

"On Parallel Processing Systems: Ahmdal's Law Generalized and Some General Results on Optimal Design" included here in the Appendix of this report.

A third major thrust which was begun toward the end of this contract period was to study the behavior of asynchronous distributed processing in a moderately tightly coupled parallel processing system. In particular, the notion of decomposing a job into a number of parallel tasks, each one of which runs on a separate processor, but each one of which is not totally independent of the others was analyzed in some preliminary studies. There are two ways to run such a system: asynchronously (the method we favor) and synchronously (a more conservative approach). We have compared the maximum improvement available for asynchronous operation versus synchronous operation in the paper by Felderman and Kleinrock "An Upper Bound on the Improvement of Asynchronous vs. Synchronous Distributed Processing" and found that the gains were clearly predictable and of some significance. Furthermore, we analyzed the details of a specific two-processor system in the paper by Kleinrock "On Distributed Systems Performance" and gave an exact analysis of the behavior of this system, demonstrating the speedup possible in such an environment. Both these papers are included in the Appendix.

In addition a number of studies involving other aspects of parallel processing systems was studied and reported upon. For example we determined the algorithms for performing optimal parallel merging and sorting using a limited number of processors in an efficient fashion. We also took advantage of broadcast communications in such sorting algorithm problems. Certain aspects of load sharing in limited access distributed systems were studied for a collection of loosely coupled parallel processors. Moreover, we studied the behavior of load sharing among a set of processors in a broadcast network; we found that if one balances the most heavily loaded processor with the most lightly loaded processor, significant gains can be had and it takes very few such pairs to be matched before the majority of the improvement is achieved; this occurs long before all processors are matched to share load thereby resulting in a very efficient load sharing at very low overhead. A related load sharing and load balancing study looked at a processing system as a "field" of processors which dynamically share load based on the immediate backlog; here again the effect of load sharing was shown both analytically and simulation-wise to be extremely effective.

Our second major area of research was in the field of distributed systems. This area covers a variety of systems models including the problems of multiaccess in broadcast communications systems, distributed database systems, and distributed search methods. Basically we have developed a number of advanced multiaccess schemes whose performance is found to be quite efficient and whose analyses have been carried out to give excellent predictions of their performance. These include schemes such as CSMA, random polling, hierarchical access schemes (which use random access in the lightly loaded portion of a network and reserved bandwidth in the more heavily used portions of a network), multiaccess in bus networks which approach gigabit speeds (thereby changing the latency vs. bandwidth bottleneck tradeoff), an analysis of access systems under dynamically changing loads, and the effect of the overhead of switching in some of these multiaccess systems. The general flavor here was to provide more effective analytic tools to be able to predict the performance of a variety of multiaccess schemes in a number of different scenarios.

We have also looked at distributed database systems using what is called optimistic concurrency control; specifically we are concerned with the performance of such systems as compared to the conservative locking schemes which are used in the more classical database control systems. We provided an analysis and simulation of optimistic concurrency control schemes and identified their effectiveness in a variety of different configurations. Lastly, in the area of distributed systems we developed and analyzed a tree search algorithm using parallel processors.

The third major area of our research involved distributed communications. This is a very rapidly growing area as we move into the domain of gigabit per second networks and the paper by Kleinrock entitled "ISDN - The Path to Broadband Networks" is included in the appendix of this report as a summary of some of out thinking and findings in this area. We are further looking into the details of congestion control in the LAN-to-LAN interconnect problem, and have made some preliminary studies of wavelength division multiplexing in optical networks. This discussion of wavelength division multiplexing has also been extended to the area of metropolitan area networks.

Overall our research progress has advanced in these many domains with considerable success in providing an understanding of the underlying behavior of parallel and distributed systems. We have created a number of evaluation tools for understanding these systems, we have studied the ways in which the algorithms and architectures behave with respect to each other and their impact upon each other, we have developed load balancing techniques and distributed database techniques along with their analyses, and have in general advanced the state of the art in order to provide a variety of tools, techniques, experimental measurements and implementations for the understanding of the behavior of parallel systems.

PSL PUBLICATIONS
DARPA CONTRACT

Computer Science Department
University of California, Los Angeles
Professor Leonard Kleinrock

July 1, 1987 - June 30, 1992

CHAPTERS IN BOOKS

1. Kleinrock, L., "Performance Evaluation of Distributed Computer-Communication Systems", Chapter 1, *Queueing Theory and Its Applications,* O.J. Boxma and R. Syski (Eds.), North-Holland Elsevier Science Publishing Company, Inc., pp. 1-57, September 1988.

2. Gerla, M. and L. Kleinrock, "Flow Control Protocols", in *Computer Network Architectures and Protocols (Second Edition),* Kenneth Schubach (Ed.), Plenum Publishing Corporation, New York, New York, pp. 265-320, May 1989.

Ph.D DISSERTATIONS

1. Huang, J. H. "On the Behavior of Algorithms in a Multi-Processing Environment", October 1988.

2. Green, Joseph "Load Balancing Algorithms in Computer Networks", October 1988.

3. Mehovic, Farid "Performance Modeling of Concurrency Control", March 1989.

4. Korfhage, Willard R. "Distributed Systems and Transient Processors", August 1989.

5. Lin, Tzung-I "Performance Analysis of Finite-Buffered Multistage Interconnection Networks with Various Switching Architectures," December 1990.

6. Felderman, Robert E. "Performance Analysis of Distributed Processing Synchronization Algorithms," June 1991.

M. S. THESES

1. Horng, Ming-Yun "An Analytic Model for Packet Flow in a Boolean N-Cube Interconnection Network", December 1987.

2. Schooler, Eve M. "Distributed Debugging in a Loosely-Coupled Processing System", February 1988.

3. Lin, Tzung-I. "An Analysis of State Restoration in Distributed Systems", March 1988.

4. Baker, Rusti "Distributed Simulation in the BBL Multicomputer System" January 1989.

5. Harinarayan Venkatesh, "Load Sharing In Limited Access Distributed Systems", 1990.

6. Chang, Chialin "Performance of LCFS Queueing Systems with Impatient Customers", June 1991.

PUBLISHED PAPERS IN PROFESSIONAL AND SCHOLARLY JOURNALS

1. Kleinrock, L. and H. Levy, "On the Behavior of a Very Fast Bidirectional Bus Network", *Proceedings of the IEEE International Conference on Communications (ICC '87), Vol. 3, No. 3,* Seattle, Washington, June 7-10, pp. 1419-1426, 1987.

2. Felderman, R.E., "Extension to the Rude-CSMA Analysis", Correspondence item, *IEEE Transactions on Communications, Vol. COM-35, No. 8,* pp. 848-849, August 1987.

3. Gerla, M. and L. Kleinrock, "Congestion Control in Interconnected LAN's", *IEEE Network, Vol. 2, No. 1,* pp. 72-76, January 1988.

4. Ferguson, C. and R. Korf, "Distributed Tree Search and its Application to Alpha-Beta Pruning", *AAAI '88 Proceedings, Vol. 1,* pp. 128-132, August 1988.

5. Kleinrock, L. and Hanoch Levy, "The Analysis of Random Polling Systems" *Journal of Operations Research, Vol. 36, No. 5,* pp. 716-732, September-October 1988.

6. Felderman, R., E. Schooler, and L. Kleinrock, "The Benevolent Bandit Laboratory: A Testbed for Distributed Algorithms", *IEEE Journal on Selected Areas in Communications, (JSAC), Vol. 7, No. 2 (ISSN 0733-8716),* pp. 303-311, February 1989.

7. Akavia, G. and L. Kleinrock, "Hierarchical Use of Dedicated Channels", *Performance Evaluation, An International Journal, Vol. 9, No. 2,* pp. 135-142, April 1989.

8. Kleinrock, L. and W. Korfhage, "Collecting Unused Processing Capacity: An Analysis of Transient Distributed Systems" reprinted from, *Proceedings of the 9th International Conference on Distributed Computer Systems,* San Diego, June 5-9, 1989, for, *The Computer Society of the IEEE,* pp. 482-489, June 1989.

9. Felderman, R. and L. Kleinrock, "An Upper Bound on the Improvement of Asynchronous Versus Synchronous Distributed Processing", *The Society for Computer Simulation,* Distributed Simulation Conference 1990, David Nicol (Ed.), pp. 131-136, January 1990.

10. Bannister, Joseph and M. Gerla, "Design of the Wavelength-Division Optical Network", presented at *ICC'90,* Vol. 3, Sec. 323.2.1, pp. 962-967, Atlanta, GA., April 16-19, 1990.

11. Huang, J-H. and L. Kleinrock, "Optimal Parallel Merging and Sorting Algorithms Using

$\sqrt{N}$ Processors Without Memory Contention", *Parallel Computing, Vol. 14, No. 1,* pp. 89-97, May 1990.

12. Bannister, J.A., L. Fratta, and M. Gerla, "Routing in Large Metropolitan Area Networks Based on Wavelength - Division Multiplexing Technology" presented at the NATO Advanced Research Workshop on *Architecture and Performance Issues of High-Capacity Local and Metropolitan Area Networks,* Sophia Antipolis, France, June 25-27, 1990.

13. Kleinrock, L. "On Distributed Systems Performance", presented at the *7th ITC Specialist Seminar, Adelaide, 1989, Proceedings of the ITC Specialist Seminar, Paper No. 3.2,* September 1990.

14. Kleinrock, L. and H. Levy, "On the Behavior of a Very Fast Bidirectional Bus Network", *IEEE Transactions on Communications,* Vol. 38, No. 10, pp. 1854-1862, October 1990.

15. Rosenberg, C., R. Mazumdar and L. Kleinrock, "On The Analysis of Exponential Queueing Systems with Randomly Changing Arrival Rates: Stability Conditions and Finite Buffer Scheme With a Resume Level", *Performance Evaluation,* Vol. 11, No. 4, pp. 283-292, November 1990.

16. Huang, J.H. and L. Kleinrock, "Distributed Selectsort Sorting Algorithms on Broadcast Communication Networks," *Parallel Computing,* Vol. 16, No. 2 & 3, pp. 183-190, December 1990.

17. Felderman, R., and L. Kleinrock, "Two Processor Time Warp Analysis: Some Results on a Unifying Approach," *Advances in Parallel and Distributed Simulation,* Proceedings of the SCS Multiconference, Anaheim, CA, Simulation Series, Vol. 23, No. 1, pp. 3-10, January 23-25, 1991.

18. Kleinrock, L. "ISDN - The Path to Broadband Networks", invited paper for *Proceedings of the IEEE,* Vol. 79, No. 2, pp. 112-117, February 1991 Special Issue on ISDN.

19. Harinarayan, V. and L. Kleinrock, "Load Sharing in Limited Access Distributed Systems," *1991 ACM Sigmetrics,* Conference on Measurement and Modeling of Computer Systems, May 21-24, 1991, San Diego, CA.

20. Kleinrock, L. and J. H. Huang, "On Parallel Processing Systems: Amdahl's Law Generalized and Some Results on Optimal Design", invited paper for *IEEE Transactions on Software Engineering,* Special issue on Performance Evaluation Methodology, Vol.18, No.5, May 1992, pp.434-447.

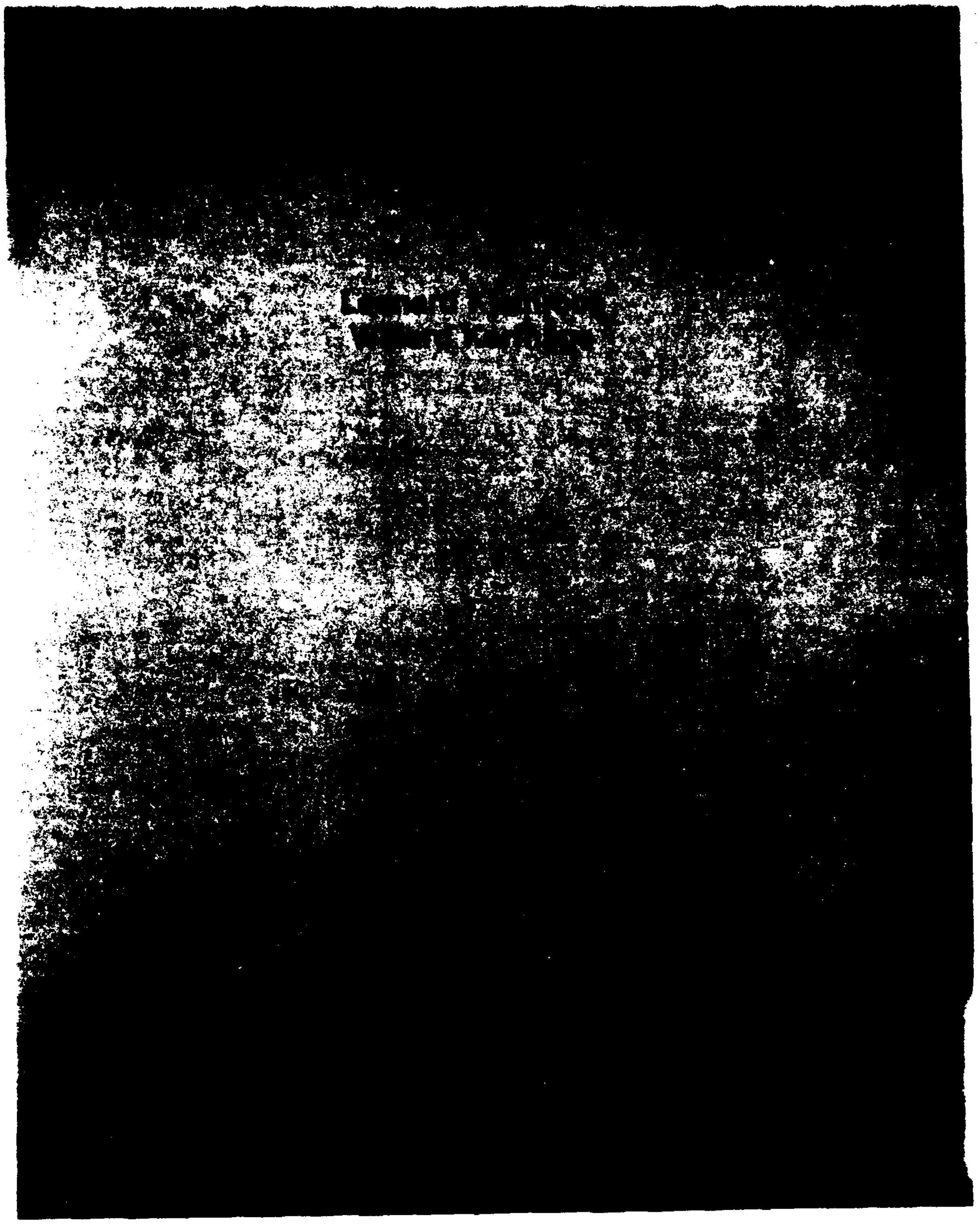
21. Kleinrock, L. and F. Mehovic, "Poisson Winner Queues," *Performance Evaluation,* Vol. 14, No. 2, 1992, pp. 79-101.

PAPERS ACCEPTED FOR PUBLICATION

1. Levy. H. and L. Kleinrock, "Polling Systems with Zero Switch-Over Periods: A General Method for Analyzing the Expected Delay," accepted for publication in *Performance Evaluation.*

2. Huang, J.H. and L. Kleinrock, "Throughput Analysis and Protocol Design for CSMA and BTMA Protocols under Noisy Environment," accepted for publication in the *IEE Proceedings, Part I* UK, April 1991.

APPENDIX



Collecting Unused Processing Capacity: An Analysis of Transient Distributed Systems

Leonard Kleinrock and Willard Korfhage*
Computer Science Department
University of California, Los Angeles

Abstract

Distributed systems frequently have large numbers of idle computers and workstations. If these could be harnessed, then considerable computing power would be available at low cost. We analyze such systems using a very simple model of a distributed program (a fixed amount of work) to see how the use of transient processors affects the program's service time.

1 Networks of Transient Processors.

Networks of computers are fairly common in business and research environments throughout the world. Originally motivated by a desire to ease data and device sharing, many networks have grown in speed and sophistication to the point that distributed processing can be performed on them. These networks vary in size from a handful of personal computers on a low-speed network, to thousands of workstations and a variety of larger machines on a high-speed, fiber-optic network. A typical example is that of workstations on a high-speed, local area network in a research laboratory. Not only are there many machines, well connected by the network, but the users are likely to demand more and more computing power.

On these networks, we often have the situation that many of the personal computers and workstations are sitting idle, waiting for their users, and thus being wasted. If we could recover this wasted time for useful processing, then we would have considerable computing power available to us at low cost [6]. We refer to these processors, which are sometimes busy and sometimes not, as *transient* processors.

This situation has analogies to time-sharing. In the past, institutions had one large resource, a mainframe computer, that was shared by many users. As networks of workstations develop, the largest computing resource is no longer a single machine, but is instead the aggregate computing power of the workstations. In the same manner that it has been possible for many people to share a single machine, by using the idle time of one person to run the program of another, so it is now possible to share a network of workstations for large, distributed computations.

Whether this is technically feasible or not depends on a variety of factors, such as the characteristics of the communications medium, the characteristics of the computers, and the statistical characteristics of the user population. Mutka and Livny [13], and Nichols [14] have shown that under at least some circumstances, this is very practical and useful.

From an analytical point of view, we would like to have a queueing model of a network of transient processors executing distributed programs. In this paper we take a first step toward this end by analyzing the service time for a very simple model of a distributed program (a fixed amount of work) to see how the use of transient processors affects what would otherwise be a deterministic service process.

2 The Model.

Assume that we have a network of M identical processors, and we wish to run a program that will require a total of W seconds of work. In general, a program consists of multiple stages of work, each of which must be completed before the start of the next. For this paper we assume that the program has only one stage of work, and furthermore, we assume that the work in any stage is infinitely divisible, and therefore is always spread evenly among the available processors.

Each processor has a capacity of one second of work per second. A processor alternates between a *non-available period*, when someone is using it, and an *available period*, when it is sitting idle. We assume that the length of non-available periods is randomly distributed with mean t_n and variance σ_n^2, and that the length of available periods is also randomly distributed from a (possibly different) distribution with mean t_a and variance σ_a^2. We wish to run our program on the network of processors, using processors while they are in their available periods. The finishing time of the program is given by a random variable $\tilde{f}$, with probability distribution $f(t)$, average $\bar{f}$, and Laplace transform $F^*(s)$. The purpose of this paper is to find

*This work was supported by the Defense Advanced Research Projects Agency under contract MDA 903-82-C0064, Advanced Teleprocessing Systems, and contract MDA 903-87-C0663, Parallel Systems Laboratory.

$f(t)$, or, in broader terms, to examine the potential use of currently wasted cycles. To do so, we examine a related function, $w(u\,|\,t)$, the probability density function of the amount of work, u, completed by time t, which has mean $\overline{W_t}$ and variance $\mathrm{Var}[W_t]$. Then, for a given W, $f(t)$ is the distribution of the time for the completed work to *first* reach W, a point also known as the *first passage time*.

If the amount of work that we wish to do is small, relative to t_a and t_n, then the finishing time is highly dependent upon the states of the processors when the program arrives. If, for example, we have a small amount of work to do on a single processor, either the processor is available with no delay, or the processor is non-available, and we must wait, on average, t_n seconds (assuming exponentially distributed non-available periods) before we can even start the work. If t_n is not small relative to W, then state of the processor when the program arrives strongly affects the finishing time distribution. In this paper, we use two techniques to mitigate the effect of this on the analysis. The single processor models make assumptions about the time that the program arrives (either in an available period or at the beginning of a non-available period). The multiprocessor model assumes that W is large relative to t_a and t_n, so the effect of initial conditions is negligible. Work is underway for the situations where W is not large.

Our analysis ultimately allows for arbitrary distributions of the lengths of available and non-available periods, but in our examples, we assume exponential distributions. Using the average available and non-available times measured by Mutka and Livny, the examples in this paper are generally based on the following parameters: $t_a = 91$ minutes, $t_n = 31.305$ minutes, $W = 10^3$ minutes for single processor examples, and $W = 10^4$ minutes for multiprocessor ($M = 100$) examples. We choose W large relative to t_a and t_n for the reasons mentioned in the previous paragraph.

We ignore communications overhead and task precedence issues in this paper, and assume that our program can use all available processors at any given time. The results of this paper, then, provide upper (i.e. optimistic) bounds on the best performance achievable in this situation.

In the remainder of this paper, we first discuss previous work by others, then analyze the problem for 1 processor. We find expressions for $w(u\,|\,t)$ and $f(t)$ in the single processor case. We then use $w(u\,|\,t)$ to find $f(t)$ for M processors and compare it to the single processor case.

3 Previous Work.

A single transient processor can be modeled in a variety of ways: as a priority queue, as a queue with vacations, as an unreliable system, or as a cumulative, alternating renewal process. Of these methods, we choose that last because it provides the asymptotic distribution of $w(u\,|\,t)$ in a very simple form and for arbitrary available and non-available period distributions. We now briefly discuss the other alternatives.

3.1 Preemptive Priority Queues.

We can model a transient processor using a preemptive priority queueing system with two classes. The high priority class represents non-available periods that interrupts the service of distributed programs, represented by the jobs in the low priority class. Such systems do not completely model a transient processor because high priority jobs continue to arrive while a high priority job is in service, and this represents a queueing of non-available periods. If we do not object to this, and if we assume that both available and non-available periods arrivals are from (different) Poisson processes, then the Laplace transform of the "completion time" (our $F^*(s)$) is known [8], and from that we can get its mean and variance.

3.2 Queues with Vacation.

Another model of a transient processor is a queueing system in which the server goes on vacations. In particular, we require that the vacations occur randomly and preempt any customer in service. If the available periods are exponentially distributed, Gaver [2] provides an analysis. For non-exponential available periods, Federgruen and Green [5] have analyzed such queues.

3.3 Unreliable Systems.

Reliability analysis concerns itself with the availability of a system over time, and some work has been done to find the distribution of cumulative availability (the cumulative amount that a system is available over time). This corresponds exactly to accumulation of work in a network of transient processors. For a system with two states (available and non-available) Donatiello and Iyer [4] find the transform of the cumulative availability, and they derive a closed-form expression when the time in each state is exponentially distributed. Using their results, we can find the moments of the amount of work done in a given amount of time for any specific available and non-available period distributions. However, we were unable to derive general results in terms of the distributions' moments, and therefore we turned to the cumulative, alternating renewal analysis described later in this paper. Although, their results do not directly provide the distribution of the first passage time, the transform of this may be obtained using techniques from the cumulative, alternating renewal analysis.

De Souza e Silva and Gail [3] discuss the calculation of cumulative availability in systems which can be modeled as homogeneous Markov chains. They find a general method for calculating cumulative availability, and further find good techniques for numerical evaluation of their method. The SAVE (System Availability Estimator) program [7] implements their method. For numerical, not approximate analytic, results, this is an excellent approach.

4 Results for One Processor.

We use two methods for deriving for the finishing time of a program on one processor. The first analysis provides the distribution of the first passage time with some restrictions on the distributions of the available and non-available periods (see section 4.1); however, it does not yield the distribution of the amount of work done over time. The second analysis, using a cumulative, alternating renewal process, provides us with the distribution of $w(u\,|\,t)$ for arbitrary available and non-available period distributions, but only with the transform of the first passage time distribution. Given $w(u|t)$ from the second analysis, we can then develop a model for M processors.

We may easily derive the averages $\overline{W_t}$ and $\overline{T}$. In a long period of time, the processor spends, on average, a fraction $t_a/(t_a+t_n)$ of its time in available periods. Thus in t seconds, the average amount of work completed is t times this fraction, or

$$\overline{W_t} = \frac{t_a}{t_a+t_n}\,t. \qquad (1)$$

Because the processor completes, on average, $t_a/(t_a+t_n)$ seconds of work per second, the reciprocal of this is the number of seconds it takes to complete one second of work. Multiplying this by W we find the average first passage time,

$$\overline{T} = \frac{t_a+t_n}{t_a}\,W. \qquad (2)$$

If we wish to account for multiple processors, the average amount of work done per time period is increased by a factor of M, the number of processors, and the first passage time is likewise decreased by a factor of M, giving us:

$$\overline{W_t} = \frac{t_a}{t_a+t_n}\,Mt. \qquad (3)$$

and

$$\overline{T} = \frac{t_a+t_n}{Mt_a}\,W \qquad (4)$$

As we will find, this simple analysis accurately characterizes the system when our program takes a long time relative to the average length of the available and non-available periods.

4.1 Direct Analysis.

Assume, for the moment, that we have only one processor ($M=1$). If our program starts when the processor is available, as shown in Figure 1, it will finish at time $W+\hat{w}$, where $\hat{w}$ is the *additional* (wasted) time the program spends in the system because the processor is sometimes in non-available periods.

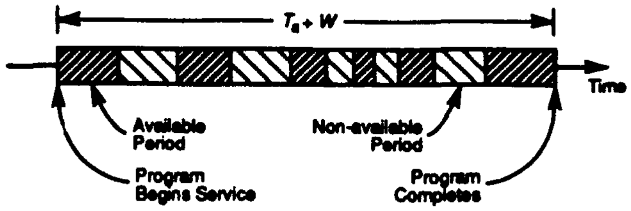


Figure 1: Time for one processor to finish W seconds of work.

Because the non-available periods are independent and identically distributed random variables with exponential distributions, $P_{\hat{w}}(t)$, the probability distribution function of the additional time in non-available periods, given that we have i such periods, is an Erlang distribution; that is

$$\frac{dP_{\hat{w}}(t|i)}{dt} = \frac{(1/t_n)(t/t_n)^{i-1}}{(i-1)!}e^{-t/t_n} \qquad (5)$$

The number of non-available periods that "arrive" during W seconds has a Poisson distribution with rate $1/t_a$. Thus, unconditioning on this number, we get that the density of $\hat{w}$ is

$$\frac{dP_{\hat{w}}(t)}{dt} = \qquad (6)$$
$$\begin{cases} e^{-W/t_a}u_0(t) & \text{if } t=0 \\ \sum_{i=1}^{\infty}\frac{(1/t_n)(t/t_n)^{i-1}}{(i-1)!}e^{\frac{-t}{t_n}}\;\frac{(W/t_a)^i}{i!}e^{\frac{-W}{t_a}} & \text{if } t>0 \end{cases}$$

where $u_0(t)$ is a unit impulse at $t=0$. For $t>0$, this may be further reduced to

$$\frac{dP_{\hat{w}}(t)}{dt} = \frac{1}{t_n}\sqrt{\frac{tW}{t_a t_n}}e^{-t/t_n}e^{-W/t_a}I_1(2\sqrt{\frac{tW}{t_a t_n}}) \qquad (7)$$

where $I_1(z)$ is the modified Bessel function of the first kind of order 1. Figure 2 shows the distribution of the first passage time using $t_a = 91$ and $t_n = 31.305$, and $W = 10^3$.

To find the mean and variance of this distribution, it simplifies the analysis to model the sequence of non-available periods using a series-parallel queueing server, shown in Figure 3, and then use known results for such servers. In this model, each of the infinite number of sub-servers (the numbered circles in the figure) has the same distribution as a single non-available period, and, in fact, represents a non-available period. We adjust the transition probabilities p_i so that the number of sub-servers a program passes through has the same distribution as the number of non-available periods arriving in W seconds. Upon entering the series-parallel server, the job immediately leaves the server with probability p_0, or the job enters sub-server

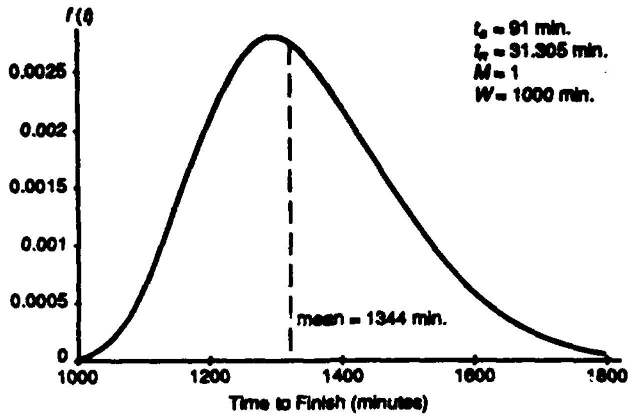


Figure 2: First Passage Time Density for Direct Analysis.

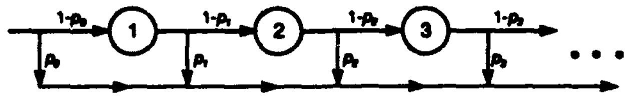


Figure 3: Series-parallel model of extra time needed to complete W seconds of work.

1 with probability $1-p_0$. Then each time a job leaves a sub-server i, a similar choice is made between leaving the series-parallel server (with probability p_i) or continuing on to the next sub-server (with probability $1-p_i$). For our purposes, we adjust the probabilities p_i so that the number of sub-servers passed through has a Poisson distribution. To compute these p_i, we note first that $p_0 = e^{-t_a W}$. For notational convenience, let $\mathcal{P}_\lambda(i) = \frac{\lambda^i}{i!}e^{-\lambda}$, the i^{th} term of a Poisson distribution with parameter λ. Next, we see that $(1-p_0)p_1 = \mathcal{P}_{W/t_a}(1)$, and we immediately have that

$$p_1 = \frac{\mathcal{P}_{W/t_a}(1)}{1-\mathcal{P}_{W/t_a}(0)}.$$

It may be proved by induction that

$$p_i = \frac{\mathcal{P}_{W/t_a}(i)}{1-\sum_{j=0}^{i-1}\mathcal{P}_{W/t_a}(j)}. \tag{8}$$

Using these p_i's, and the expression in Kleinrock [10] for $\hat{W}^*(s)$, the Laplace transform of the density of the time required to pass through a series-parallel server, we get

$$\hat{W}^*(s) = p_0 + \sum_{i=1}^{\infty} \frac{(W/t_a)^i}{i!}e^{-(W/t_a)}\left(\frac{1/t_n}{s+1/t_n}\right)^i \tag{9}$$

which, after some manipulation, becomes

$$\hat{W}^*(s) = \mathcal{P}^*_{W/t_a}\left(\frac{1/t_n}{s+1/t_n}\right) \tag{10}$$

where $\mathcal{P}^*_\lambda(z) = e^{\lambda(z-1)}$ is the z-transform of a Poisson distribution with parameter λ.

We multiply this by e^{-Ws} to account for the W seconds that we are required to work, and finally find the transform for the distribution of time to finish W seconds of work, i.e.,

$$F^*(s) = e^{-Ws}e^{(W/t_a)((1/t_n)/(s+1/t_n)-1)}. \tag{11}$$

Taking derivatives, we find that the mean time required to finish W seconds of work is

$$\overline{f} = W\frac{t_a+t_n}{t_a}, \tag{12}$$

and the variance of this time is

$$\sigma_f^2 = 2\frac{t_n^2 W}{t_a} \tag{13}$$

Note that Equation 12 agrees with Equation 2. This mean and variance may also be derived by viewing the distribution as a constant plus a random (Poisson-distributed) sum of Erlang random variables, and using the well-known formulas for the mean and variance of a random sum of random variables [10].

We can modify this analysis for non-exponential non-available time distributions, but it still requires exponential available times so that the number of non-available periods in W seconds has a Poisson distribution. If we wish to allow arbitrary available period distributions as well, then we need a more sophisticated method of counting non-available periods, and this leads to analyzing the situation as a renewal process.

4.2 Analysis as a Cumulative, Alternating Renewal Process

To allow arbitrary distributions for the available and non-available periods, we follow the presentation of Cox [1] in his treatment of cumulative, alternating renewal processes. As before, the single processor has a capacity of one second of work per second, and we wish to find the distribution of the time required for this processor to finish W seconds of work. Let X_i' be the duration of the i^{th} non-available period, and X_i'' be the duration of the i^{th} available period, as shown in Figure 4. Let the renewal points for our alternating renewal process be the beginning of the non-available periods, shown as heavy dots in the figure; the time between renewal points is then $X_i = X_i' + X_i''$. This has mean $E[X] = t_a + t_n$ and variance $Var[X] = \sigma_a^2 + \sigma_n^2$. We assume that $t = 0$ occurs at the beginning of a renewal period.

To form a cumulative process from this, define W_i to be the amount of work completed in each renewal period: $W_i = X_i''$, with mean t_a and variance σ_a^2. Let Z_t be the sum of all the available time up to time t, excepting that in the current available period, if the

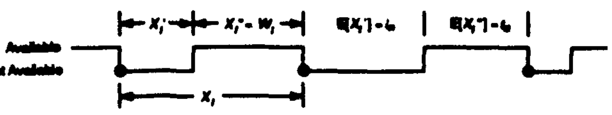


Figure 4: Cumulative Renewal Process

process is in such a period at time t.

$$\begin{aligned} Z_t &= \sum_{i=1}^{N_t} W_i \quad (N_t = 1, 2, \ldots) \qquad (14) \\ &= 0 \qquad (N_t = 0) \end{aligned}$$

where N_t is the number of renewals in $(0, t]$. Asymptotically, Z_t has the same properties as $w(u \mid t)$, the process that accumulates the true total available time up to time t; however, Z_t's analysis is more tractable than that of $w(u \mid t)$.

Cox's analysis allows the available and non-available periods to have arbitrary distributions. For t large, Z_t is the sum of many independent random variables, and it is asymptotically normal with mean

$$\mathrm{E}[Z_t] = \frac{t_a}{t_a + t_n}\, t \qquad (15)$$

and variance

$$\mathrm{Var}[Z_t] = \frac{2(\sigma_a^2 t_n^2 + \sigma_n^2 t_a^2)}{(t_a + t_n)^3}\, t. \qquad (16)$$

Using exponentially distributed available and non-available periods, these become:

$$\mathrm{E}[Z_t] = \frac{t_a}{t_a + t_n}\, t \qquad (17)$$

$$\mathrm{Var}[Z_t] = \frac{2(t_a t_n)^2}{(t_a + t_n)^3}\, t. \qquad (18)$$

As noted before, the asymptotic properties of Z_t are identical to the asymptotic properties of $w(u \mid t)$.

Comparing this to the result of the direct analysis, we note that for t equal to the mean first passage time (Eqn. 12), we have done, on average, W seconds of work, as we expected.

In his derivation, Cox, assuming that W_i is independent of X_i'' for all i, derives a double transform for $f(t)$. Unfortunately, this transform is very difficult to invert, but the asymptotic distribution of Z_t is really what we need so we can use it in the next analysis.

5 Results for M Processors

Because the amount of work done by one transient processor is the sum of a (possibly large) number of available periods, the total work done in time t is asymptotically normal with mean and variance given in Equations 17 and 18, respectively. If $t_a \ll W$ and $t_n \ll W$, then it is reasonable to use Brownian motion as a model of an M processor system. All M processors are assumed to be independent, so the amount of work done by time t is the sum of M independent, (approximately) normally distributed random variables, with mean

$$\bar{b}t = \frac{t_a}{t_a + t_n} M t = p_a M t \qquad (19)$$

and variance

$$\sigma_b^2 t = \frac{2(\sigma_a^2 t_n^2 + \sigma_n^2 t_a^2)}{(t_a + t_n)^3}\, M t. \qquad (20)$$

For exponential available and non-available distributions, Eqn. 20 becomes:

$$\sigma_b^2 t = \frac{2(t_a t_n)^2}{(t_a + t_n)^3} M t = \frac{2 p_a^2 (1 - p_a) M t}{t_n} \qquad (21)$$

where $p_a = t_a / (t_a + t_n)$.

With these as the parameters for our Brownian motion, the density of the time, t, that it takes for M

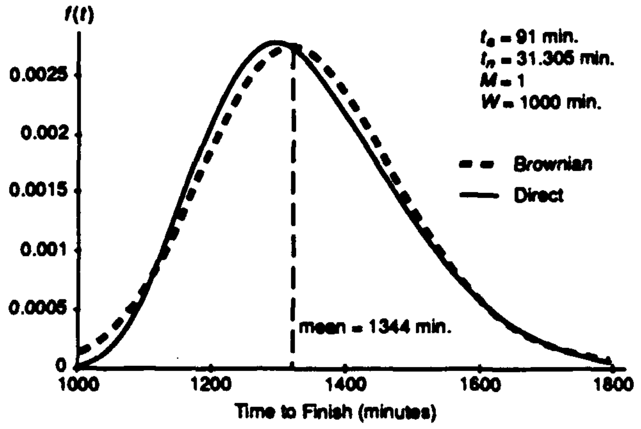


Figure 5: First Passage Time Distributions for Direct and Brownian Motion Analyses.

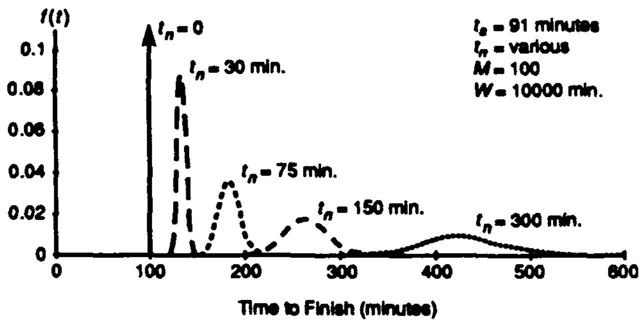


Figure 6: First Passage Time Densities for Brownian Motion Model for Various t_n.

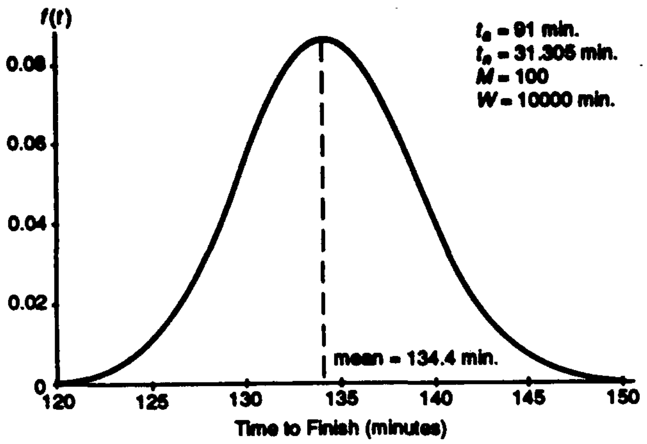


Figure 7: First Passage Time Density for Brownian Motion Analysis.

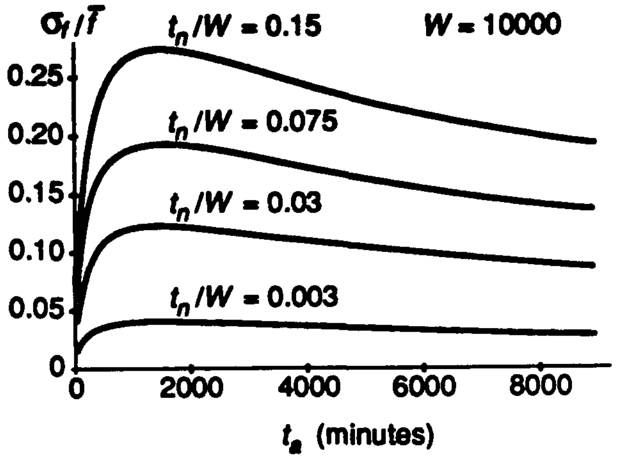


Figure 8: $\sigma_f/\bar{f}$ with various t_a and t_n.

processors to finish W seconds of work is well-known (e.g. see Karlin and Taylor [9]) and is given by:

$$f(t) = \frac{W}{\sqrt{2\pi\sigma_b^2 t^3}} \exp\left[-\frac{(W-\bar{b}t)^2}{2\sigma_b^2 t}\right] \tag{22}$$

This has mean

$$\bar{f} = \frac{W}{\bar{b}} = \frac{W}{M}\frac{(t_a+t_n)}{t_a} \tag{23}$$

and variance

$$\sigma_f^2 = \frac{W}{\bar{b}}\frac{\sigma_b^2}{\bar{b}^2}. \tag{24}$$

and for exponential distributions, the latter becomes

$$\sigma_f^2 = \frac{2W}{M^2}\frac{t_n^2}{t_a}. \tag{25}$$

Equation 22 is the main result of this paper. Note that it makes no assumptions about the distributions of the available and non-available periods, except that their variances are finite, and only the distributions' means and variances appear in the first passage time.

Note that for the case $M = 1$, this mean and variance agree with our single processor analysis of section 4.1. The first passage time densities for both the single processor analysis and the multiprocessor analysis with $M = 1$ are shown in Figure 5.

Figure 6 shows the distribution of first passage time for various t_n with $t_a = 91, M = 100$, and $W = 10^4$. Using the $t_a = 91$ and $t_n = 31.305$, our job of 10^4 minutes would take about a week to run on a single, dedicated processor. When run on a network of 100 transient processors, it would take 134.06 minutes, or about 2.25 hours. The distribution of this first passage time is shown in Figure 7. Note that although $W \gg t_a$ and $W \gg t_n$, we have that W/M, the finishing time if the processors were fully dedicated to the program, is close to the average length of an available period, and it is remarkable that the curve is still so symmetrical.

It is useful to examine the ratio of σ_f to $\bar{f}$, namely:

$$\frac{\sigma_f}{\bar{f}} = \frac{\sqrt{\sigma_b^2}}{\sqrt{\bar{b}W}} \tag{26}$$

to see what happens to the distribution as the parameters change. For exponential distributions this becomes

$$\frac{\sigma_f}{\bar{f}} = \sqrt{\frac{2t_n}{W}}\frac{\sqrt{t_n/t_a}}{1+t_n/t_a}, \tag{27}$$

and it is this equation that we examine in more detail. Figure 8 plots Eqn. 27 for fixed t_n and W, and varying t_a. Note that in this figure, $t_n = 30$ minutes is the lowest line and $t_n = 1500$ minutes is the uppermost line.

Because we assumed $t_n \ll W$, this ratio tends to be less than 1, reaching a peak when $t_a = t_n$. If we fix t_n/W and let t_n/t_a go to infinity (which implies $t_a \to 0$), the ratio goes to 0. We explain this by noting that for small t_a, it takes very many available–non-available cycles before the work is finished. The law of large numbers insures that the first passage time distribution, which is the sum of these many periods, will then be tight about its mean.

If, on the other hand, we let $t_a \to \infty$, the ratio of the standard deviation to the mean goes to zero once again. When t_a is large relative to t_n, the non-available periods become negligible, as if the processors are always available. Again, the first passage time distribution becomes very tight about its mean because non-available time periods add little variability to the finishing time.

Given that the first passage time distribution is tight about its mean, (i.e. $t_a \gg t_n$, or $t_a \ll t_n$, or $W \gg t_n$), it may be accurate enough to consider the distribution as an impulse at the mean finishing time (in the spirit of the law of large numbers). Using the previous example again, we find $\sigma_f^2 = 21.53$, and approximating $f(t)$ as a normal distribution (discussed below), we find that 90% of the time, programs requiring 10^4 minutes of work will finish within 7.6 minutes of the 134.4 minute mean finishing time.

The central-limit theorem says that $f(t)$ will tend toward a normal distribution when many available—non-available periods occur before the program completes (i.e. $W \gg t_a$ and $W \gg t_n$). To approximate the first passage time, we use a normal distribution with the same mean and variance as the first passage time distribution:

$$\hat{f}(t) = \frac{1}{\sqrt{2\pi\sigma_f^2}} e^{-(t-\bar{f})^2/(2\sigma_f^2)} \qquad (28)$$

When the mode of the first passage time distribution is close to its mean, a normal distribution well approximates the first passage time. The mode is:

$$t_{mode} = \frac{1}{2}\sqrt{\frac{9(\sigma_b^2)^2}{\bar{b}^4} + \frac{4W^2}{\bar{b}^2}} - \frac{3\sigma_b^2}{2\bar{b}^2}. \qquad (29)$$

Comparing this to the mean, we find that the percentage difference between the two is approximately $3\sigma_b/2\bar{b}W$, which, as we would expect, shrinks as W grows.

6 Conclusion

We have analyzed a network of transient processors, and determined the probability density of the length of time it takes to finish a fixed amount of work. The main result for an M processor network is given in Equation 22, and it is valid for general available and non-available period distributions. Simulations confirm that Brownian-motion-with-drift is an accurate model of system performance under the assumptions given above. With large programs that run for a long time relative to the length of available and non-available periods, the central limit-theorem applies, and the Brownian-motion-with-drift model remains good regardless of the distributions of the available and the non-available periods. Under these assumptions, the distribution of finishing time will be very tight about its mean, and is well approximated by a normal distribution.

It does remain to account for communication overhead and precedence relationships, but it is likely that these can be accommodated, or at least approximated, within the model.

The analysis in this paper has not examined the effect of multiple programs in the network. We may now use the first passage time distribution (Eqn. 22), as the service time in a queueing system that represents the network. If each job gets the whole network and they must queue, then a G/G/1 queue is a good model. If all the programs in the network share the processors equally, then we could model the network as an M/G/1 processor-sharing system. The analysis of such systems remains for a future paper.

It is well known that, for a given total processing capacity, the average response time is shortest if we use one large processor rather than many small processors [11]. From this perspective, the trend toward individual workstations is a curious one. However, this result assumes that each program executes on only one processor. If we distribute the program over *all* the small processors, then we may recover, at least partially, the response time advantages of a large, central system, while retaining the advantages of individual workstations.

References

[1] D. R. Cox. *Renewal Theory*. Methuen and Co., Ltd., London, science paperbacks edition, 1962.

[2] Jr. D. P. Gaver. A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society*, B24:73, 1962.

[3] Edmundo de Souza e Silva and H. Richard Gail. Calculating Availability and Performability Measures of Repairable Computer Systems Using Randomization. *Journal of the ACM*, 36(1):171–193, January 1989.

[4] Lorenzo Donatiello and Balakrishna R. Iyer. Closed-Form Solution for System Availability Distribution. *IEEE Transactions on Reliability*, R-36(1):45–47, April 1987.

[5] A. Federgruen and L. Green. Queueing Systems with Service Interruptions. Research Working Paper 84-5, Columbia University, 1984.

[6] Robert Felderman, Eve Schooler, and Leonard Kleinrock. The Benevolent Bandit Laboratory: A Testbed for Distributed Algorithms. *IEEE Journal on Selected Areas in Communications*, 7(2):303–311, February 1989.

[7] Ambuj Goyal. System Availability Estimator (SAVE) User's Manual Version 2.0 (External). Technical Report RC 12517 (No. 56267), IBM Watson Research Center, February 1987.

[8] Daniel P. Heyman and Matthew J. Sobel. *Stochastic Models in Operations Research, Volume 1: Stochastic Processes and Operating Characteristics*. Series in Quantitative Methods for Management. McGraw Hill, 1982.

[9] Samuel Karlin and Howard M. Taylor. *A First Course in Stochastic Processes*. Academic Press, second edition, 1975.

[10] Leonard Kleinrock. *Queueing Systems, Volume 1: Theory*. John Wiley and Sons, 1975.

[11] Leonard Kleinrock. Distributed Systems. *Communications of the ACM*, 28(11):1200–1213, November 1985.

[12] Willard Korfhage. *Distributed Systems and Transient Processors*. PhD dissertation, University of California, Los Angeles, 1989.

[13] Matt W. Mutka and Miron Livny. Profiling Workstation's Available Capacity for Remote Execution. Computer Sciences Technical Report 697, CS Dept., Univ. of Wisconsin, May 1987.

[14] David A. Nichols. Using Idle Workstations in a Shared Computing Environment. In *Proceedings of the Eleventh ACM Symposium on Operating System Principles*, pages 5–12. ACM, November 1987.

On Parallel Processing Systems: Amdahl's Law Generalized and Some Results on Optimal Design

Leonard Kleinrock, *Fellow, IEEE*, and Jau-Hsiung Huang

***Abstract*—We model a job in a parallel processing system as a sequence of stages, each of which requires a certain integral number of processors for a certain interval of time. With this model we derive the speedup of the system for two cases: systems with no arrivals, and systems with arrivals. In the case with no arrivals, our speedup result is a generalization of Amdahl's Law. We extend the notion of "power" (the simplest definition is power = throughput/response time) as previously applied to general queueing and computer-communication systems to our case of parallel processing systems. With this definition of power we are able to find the optimal system operating point (i.e., the optimal input rate of jobs) and the optimal number of processors to use in the parallel processing system such that power is maximized. Many of the results for the case of arrivals are the same as for the case of no arrivals. A familiar and intuitively pleasing result is obtained, which states that the average number of jobs in the system with arrivals equals unity when power is maximized.**

We also model a job in a way such that the number of processors required is a continuous variable that changes continuously over time. The same performance indices and parameters studied in the discrete model are evaluated for this continuous model. These continuous results are more easily obtained, are easier to state, and are simpler to interpret than for the discrete model.

***Index Terms*—Amdahl's Law, multiprocessing, optimal design, parallel processing, power, processor efficiency, speedup, system utilization.**

I. Introduction

As parallel computing systems proliferate, the need for effective performance evaluation techniques becomes ever more important. In this paper, we study certain fundamental performance indices, namely, *speedup*, *response time*, *efficiency*, and *power*, and solve for the optimal operating point of these systems. Specifically, by maximizing "power," we are able to find the optimal input rate of jobs and the optimal number of processors to use, given a characterization of the workload.

We model a parallel processing system as a system with a single queue of waiting jobs. Our first model (in Section IV) assumes that only a single job needs to be processed. Our second model (in Section V) allows a stream of arrivals to enter the system; however, only one job may be admitted into service at a time, following a FCFS discipline, while the others wait in the queue. Both models deal with jobs as follows. While in service, the system provides a maximum of P parallel processors to work on the job. A job is modeled as a sequence of independent stages which must be processed, where the number of processors desired by the job in each stage may be different. If, for some stage, the job in service requires fewer processors than the system provides, then the job will use all that it needs and the other processors will be idle for that stage. If, for some other stage, the job in service requires more processors than the system provides, then it will use all the processors in the system (in a processor sharing fashion [10]) for an extended period of time such that the total work served in that stage is conserved. An example is given in Fig. 1 in which the total processing work required by a job is $W = 24$ s. In this example, if $P \geq 6$, then it takes 8 s to complete the job as shown in Fig. 1(a), whereas if only $P = 4$ processors are provided, then it takes 9 s as shown in

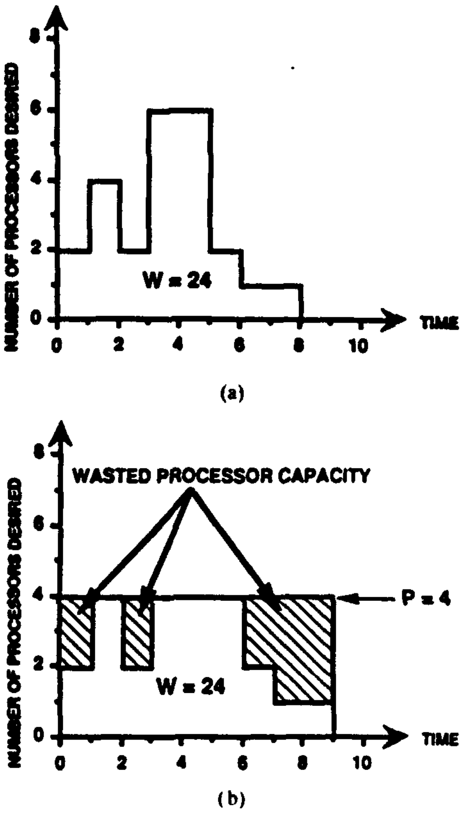


Fig. 1. Job profile. (a) Unlimited number of processors. (b) Limited number of processors ($P = 4$).

Manuscript received April 1, 1991; revised September 20, 1991. Recommended by E. Gelenbe. This work was supported by the Defense Advanced Research Projects Agency, Department of Defense under Contract MDA903-87-C-0663.

L. Kleinrock is with the Computer Science Department, University of California, Los Angeles, Los Angeles, CA 90024.

J.-H. Huang is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

IEEE Log Number 9105395.

Fig. 1(b), in which case 12 s of processor capacity are wasted.

The model described above has been highly idealized. In particular, we are neglecting some of the following important aspects of the workload. First, we do not allow general precedence relations among the tasks. Our precedence structure is equivalent to a series-parallel task graph with deterministic task service times (see [6] for the definition of the task graph model of computation). Second, we do not separately model the communication times between tasks (i.e., the interprocess communication overhead). We hasten to point out that incorporating this overhead is not simply a matter of adding additional time to each task's processing time, since such overhead only occurs when a task on one processor must pass its results to a task on a different processor; thus to properly include interprocess communication costs, one must model the way in which tasks are assigned to processors (i.e., the task partitioning problem), an assignment that we choose to neglect. Third, we ignore I/O communication overhead related to the management and execution of parallel programs. Lastly, we assume that the program structure is infinitely divisible, in that the time to execute w units of work is equal to $\max(w/P, w/P')$, where P is the number of processors that the system provides for execution of this work, and P' is the maximum number of processors that the program is able to use for this work (i.e., the parallelism for this work). These assumptions simplify our analysis and lead to idealized results.

Our workload model was first reported by us in [8]. Later, Gelenbe [6] described a very similar model, as did Sevcik [15]. Gelenbe extended his model, which he referred to as the "Activity Set Model," to include the effect of inefficient use of processors, imbalance of the workload among the processors, and interprocess communication times. Sevcik also described ways in which this idealized model could be extended to include the effect of I/O communications, overhead, and dependencies among parallel threads assigned to different processors.

For such a parallel processing system there are two performance measures which compete with each other: *processor efficiency* and *mean response time*. One can increase the processor efficiency of the system (by reducing the number of processors), but then the mean response time will also be increased. Similarly, one can lower the mean response time (by increasing the number of processors), but then the processor efficiency of the system will also be lowered. In this paper these two performance measures are combined into a single measure, known as *power*, which increases by either lowering the mean response time or by raising the processor efficiency of the system. We seek to find that number of processors which maximizes power.

Power, studied in [5], [11], and [12], was defined for a general queueing system in [12] as

$$\frac{\rho}{T/\bar{x}}$$

where ρ is defined as the system utilization, T is defined as the mean response time, and $\bar{x}$ is defined as the average service time. With this measure we see that an increase in system utilization (ρ) or a decrease in response time (T) increases the power. (Note that this normalized definition is such that since $0 \leq \rho < 1$, and since $1 \leq T/\bar{x}$, then $0 \leq$ power < 1.) The symbol "*" will be used throughout to denote variables which are optimized with respect to power. In [12] it was found that for any M/G/1 queueing system [9], power is maximized when $\bar{N}^* = 1$, where $\bar{N}$ = the average number of jobs in the system. This result says that an M/G/1 system has maximum power when on the average there is only one job in the system. This result is intuitively pleasing, since it corresponds to our deterministic reasoning that the proper operating point for a single-server system is exactly when only one job is being served in the system and no others are waiting for service at the same time. In this paper, our results also show that $\bar{N}^* = 1$ when power is maximized with respect to the job arrival rate (λ).

One might argue that power, as here defined, is an arbitrary performance measure. In response to this argument we point out that one can generalize the definition of power in a way which allows the reader to emphasize delay (or efficiency) in a variety of ways so as to match his or her needs. This issue is discussed below in Section II as well as in [5] and [12]. Moreover, other researchers have seen fit to optimize power for models similar to ours (see, for example [4]). An extensive study of power applied to computer networks is given in [5].

An alternative, and much more familiar, performance measure for parallel processing systems is *speedup*, which describes how much faster a job can be processed using multiple processors, as compared to using a single processor. Specifically, speedup is the ratio of the mean response time of a job processed by a single processor to that of a job executed in a parallel processing system with, say, P processors. Speedup and power are related and we discuss how they interact throughout this paper. Eager *et al.* [4] also discuss issues similar to those in this paper. Their focus is on estimating speedup and efficiency (for the no arrivals case only) simply from the value of the "average parallelism," which is defined as W, the total processing work required by a job, divided by the time it would take to service the job if there were an unlimited number of processors available; in Fig. 1(a) we have $W = 24$, and service time = 8, giving an average parallelism equal to 3. They also use the definition of power as we had defined in [11] and [12] and obtain the same result as we obtain in Corollary 7 below. They consider the case of deterministic workloads. Gelenbe [6] introduced an alternate model for the workload for which he also calculates speedup in the case of an infinite number of available processors. He models a job as having a random task graph in which the density of precedence relations between tasks is given by p ($0 \leq p \leq 1$); he then derives an approximation for an upper bound on the speedup; namely, $(1+p)/2p$.

II. Definitions

We have already defined the following:

P = Number of (identical) processors in the server;

W = Average number of seconds required to process a job on a single processor; and

$\bar{N}$ = Average number of jobs in the system.

Moreover, we now define the following additional quantities:

$\bar{x}(P) =$	Mean service time of a job in a P-processor system (note that the maximum mean service time is $\bar{x}(1) = W$ and that the minimum mean service time is $\bar{x}(\infty)$);
$T(\lambda, P) =$	Mean response time (queueing time plus service time) of a job in a queueing system with an input rate λ and P processors;
$\lambda =$	arrival rate of jobs;
$\rho =$	system utilization; i.e., the fraction of time when there is at least one job in the system.
$=$	$\lambda\bar{x}(P)$; and
$u(P) =$	processor efficiency in a P-processor system.

Note the difference between $u(P)$, which is the average *processor* efficiency given P processors, and ρ, which is the average *system* utilization. Whenever there is a job in the system, the system utilization is "1," but the processor efficiency need not be "1" in that case, since there may be some idle processors (i.e., it may be that the job in service does not require all the processors). Hence the system utilization is always greater than or equal to the processor efficiency. (Note that $u(1) = \rho$ for a single processor queueing system.)

Two cases regarding the number of jobs in the system are considered in this paper. Case one allows no arrivals of additional jobs (Section IV). That is, there is only one job in the system, and we are concerned with $\bar{x}(P)$, its mean service time in a P-processor system. Case two allows jobs to arrive from a Poisson process at a rate λ, and so queueing effects are considered (Section V).

For the first case, we define the (no arrivals case) *speedup* with P processors, denoted by $S_n(P)$, to be

$$S_n(P) = \frac{\bar{x}(1)}{\bar{x}(P)} = \frac{W}{\bar{x}(P)}.$$

Note that

$$1 = \frac{W}{\bar{x}(1)} \le S_n(P) \le \frac{W}{\bar{x}(\infty)}.$$

Thus it is natural for us to define the maximum value for speedup $S_{n,\max}$ as follows:

$$S_{n,\max} = \frac{W}{\bar{x}(\infty)}.$$

Furthermore, we see that $S_{n,\max}$ = average parallelism.

For the second case, we define the (arrivals case) *speedup* with P processors at system utilization ρ, denoted by $S_a(\lambda, P)$, to be

$$S_a(\lambda, P) = \frac{T(\lambda, 1)}{T(\lambda, P)}.$$

We must distinguish the processor efficiency $u(P)$ in these two cases as follows:

$u_n(P) =$	processor efficiency given P processors in the no arrivals case; and
$u_a(\lambda, P) =$	processor efficiency given job arrival rate λ and P processors in the case with job arrivals.

We now introduce the appropriate definitions of *power*, which we denote by the symbol Q (we would prefer to use the obvious notation P, but P has already been used to denote the number of processors). Let

$Q_n(P) =$	power given P processors in the no arrivals case; and
$Q_a(\lambda, P) =$	power given a job arrival rate λ and P processors in the case with job arrivals.

In this paper we are concerned mostly with power which is defined as processor efficiency divided by the mean response time.

In the case of no arrivals, the mean response time of the (single) job is simply its mean service time $\bar{x}(P)$, and so:

$$Q_n(P) = \frac{u_n(P)}{\bar{x}(P)}.$$

Clearly, power will increase by either raising the processor efficiency or by lowering the mean service time. A more general definition of power (as originally introduced in [12]) is given as

$$Q_n^{(r)}(P) = \frac{[u_n(P)]^r}{\bar{x}(P)}$$

where r is a positive real number whose value may be selected by the system designer. With this generalization, a designer may express a stronger preference for an increase in the processor efficiency at the expense of an increase in the mean service time by simply increasing the value of the parameter r (and vice-versa). Note that $Q_n(P) = Q_n^{(1)}(P)$.

In the case of job arrivals, the definition of power becomes:

$$Q_a(\lambda, P) = \frac{u_a(\lambda, P)}{T(\lambda, P)}$$

and the generalization in this case is

$$Q_a^{(r)}(\lambda, P) = \frac{[u_a(\lambda, P)]^r}{T(\lambda, P)}$$

where again r is a positive real number to be used as a degree of freedom by the system designer. Note that $Q_a(\lambda, P) = Q_a^{(1)}(\lambda, P)$.

With these definitions of power, our goal is to find the optimal number of processors to use in a parallel processing system such that power is maximized. Furthermore, in the case of job arrivals, we also seek the optimal system operating point (i.e., the optimal input rate of jobs).

The rest of this paper is organized as follows. In Section III we present two models of a job: a discrete model, and a continuous model. In Section IV we solve the case when no arrivals are allowed in the system. In this case we find the speedup of the system given P processors. We also find P^*, the number of processors which maximizes power. In Section V we solve the case when job arrivals are allowed in the system. In this case we again solve for the speedup of the system given P processors. We also find λ^* and P^*, which maximize power. One interesting result we get is that the P^* for systems with no arrivals and the P^* for systems with arrivals are equal when power is maximized; this provides a simplification in system design.

III. Workload Models

We consider both a discrete as well as a continuous model of job requirements.

A. A Discrete Job Model

Here, we model a job as containing a total of $\widetilde{W}$ tasks. Nonoverlapping subsets of these tasks are collected into *stages*, and these stages are processed sequentially (however, parallelism is exploited *within* each stage—see below). $\widetilde{W}$ is a random variable with mean W and coefficient of variation c_W[1]. We assume that the service time distribution for each task is deterministic, such that each task requires 1 s of work on a processor. For the results we seek in this paper, a job is described by specifying W and c_W along with two other vectors. The first vector is called the *fraction* vector, $\boldsymbol{f}'$, and the second vector is called the *processor* vector, $\boldsymbol{P}'$. We denote the fraction and processor vectors as

$$\boldsymbol{f}' = [f_1', f_2', f_3', \cdots, f_{n'}']$$
$$\boldsymbol{P}' = [P_1', P_2', P_3', \cdots, P_{n'}']$$

where n' is the number of stages in a job. The ith stage has the pair (f_i', P_i') associated with it. The meaning is as follows: a fraction f_i' of the total tasks in a job can use P_i' processors to concurrently process these tasks. For this definition, it is clear that

$$\sum_{i=1}^{n'} f_i' = 1.$$

The example from Fig. 1 is repeated in Fig. 2(a), where $W = 24$ and $n' = 6$. Stage 4 contains 12 tasks, and so $f_4' = 1/2$; moreover, since $P_4' = 6$ (and if $P \geq 6$), then it will take 2 s to complete stage 4. This stage-type workload model comes directly from the usual task graph model of computation [3] with deterministic task service times. The ith stage corresponds to the ith level in the computation graph.

For convenience, we may rearrange the elements in $\boldsymbol{f}'$ and $\boldsymbol{P}'$ as follows in such a way that neither the mean response time nor the processor efficiency are changed. The elements of $\boldsymbol{P}'$ are rearranged and renumbered so that its elements are nondecreasing; that is, $P_{i-1}' \leq P_i'$. The elements of $\boldsymbol{f}'$ follow the identical permutation and renumbering. We may then merge several stages with the same P_i''s into one stage simply by adding all the corresponding f_i''s. The new vectors will be denoted $\boldsymbol{P} = [P_1, P_2, \cdots, P_n]$ and $\boldsymbol{f} = [f_1, f_2, \cdots, f_n]$, where $n \leq n'$ and $P_{i-1} < P_i$. Since the system admits only one job into service at a time, it can easily be shown that this rearrangement does not affect the performance at all. The example in Fig. 2(a) has been rearranged as shown in Fig. 2(b), where the number of stages is now $n = 4$. Note that $\bar{x}(\infty) = 8$, as it was in Fig. 2(a). One can easily see that if we choose $P = 4$, then $\bar{x}(4)$ will equal 9 in this rearranged case, as was the case for Fig. 1(b).

[1]The coefficient of variation of a random variable is equal to its standard deviation divided by its mean.

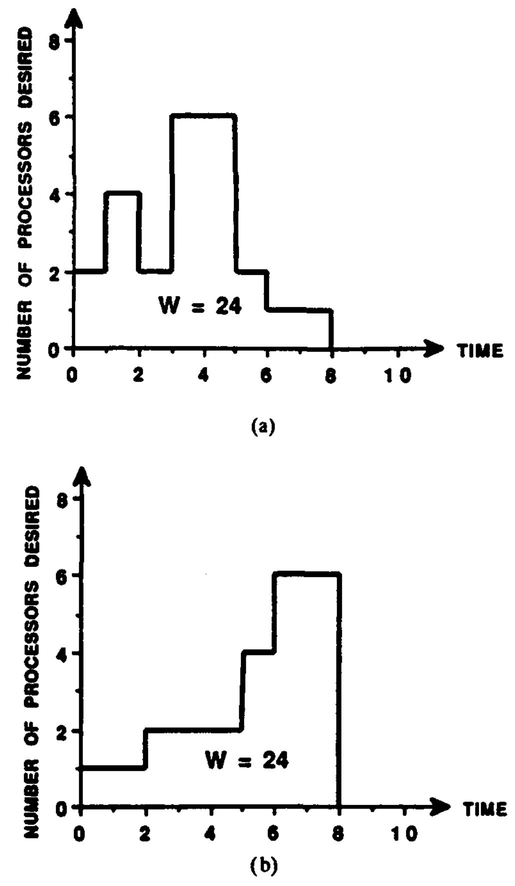


Fig. 2. Rearranging the job profile. (a) $\boldsymbol{P} = [2,4,2,6,2,1]$, $\boldsymbol{f} = [\frac{1}{12}, \frac{1}{6}, \frac{1}{12}, \frac{1}{2}, \frac{1}{12}, \frac{1}{12}]$. (b) $\boldsymbol{P} = [1,2,4,6]$, $\boldsymbol{f} = [\frac{1}{12}, \frac{1}{4}, \frac{1}{6}, \frac{1}{2}]$.

B. A Continuous Job Model

We now describe a continuous version of the above model. In this model we assume that the number of processors required by jobs is a (not necessarily discrete) nondecreasing function of time (recall the rearranging does not affect performance). That is, we permit nonintegral numbers of processors (which could correspond to cases where processors are shared among more than one job). A special model with a deterministic workload per job will be described first, and then a more general model with a random workload per job will be described.

For the special case with a deterministic workload, we define $P(t) = g(t)$, where $g(t)$ is a deterministic function, to be the number of processors that a job desires at time t $(0 \leq t \leq b)$ such that $P(b) = B$ (see Fig. 3). For such a model, the workload (seconds of work required) for each job is deterministic with value

$$W = \int_0^b P(t)\, dt.$$

Note that $b = \bar{x}(\infty)$. Moreover, if we limit the number of processors to $P (P < B)$, then A, the (shaded) area of $P(t)$ which lies above the value of P, will be flattened out and extended as a rectangle of area A and of height P beginning

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING

MAY 1992 VOLUME 18 NUMBER 5 IESEDJ (ISSN 0098-5589)

A PUBLICATION OF THE IEEE COMPUTER SOCIETY

An Upper Bound on the Improvement of Asynchronous versus Synchronous Distributed Processing*

Robert E. Felderman and Leonard Kleinrock
UCLA Computer Science Department
3732L Boelter Hall
Los Angeles, CA 90024-1596

Abstract

We use simple models of two distributed processing methods, one asynchronous, the other synchronous, to calculate the maximum potential performance gain of the former over the latter. We show, in the limit as the number of tasks grows and the number of processors increases, that the asynchronous method has an expected potential speedup over the synchronous method of no more than $\ln P$ where P is the number of processors used by each strategy.

1 Introduction

We compare two synchronization methods used in distributed processing systems and determine how much better one performs than the other. Our motivation comes from the area of Parallel Discrete Event Simulation (PDES) which has received much attention recently [Misra 1986] [Jefferson 1985]. There are several algorithms used for PDES and this paper demonstrates the potential improvement by using an asynchronous approach (e.g. Time Warp), over a synchronous technique, (e.g. time-stepped simulation). We first give an introduction to PDES, discuss briefly the two methods chosen for comparison, and then follow with our models and analysis. Readers who are unfamiliar with Discrete Event Simulation and techniques used to parallelize it are referred to [Misra 1986] [Jefferson 1985] [Peacock et al. 1979] for more details.

2 Parallel Discrete Event Simulation

Parallel Discrete Event Simulation is generally accomplished by partitioning the simulation into logical processes (LP) or *entities* each of which simulates some physical process in the system. An example is the simple queueing network shown in Figure 1. Entities in our system are the customer arrival process(A), the servers(B,C,D,E) and a final sink process(F) to collect departing customers. Each process receives messages, performs internal computations and sends messages to other processes. Each LP maintains a local clock which indicates the current time of the simulation at that entity, and a process terminates once its local or logical clock (the simulation time of the message currently being processed) has reached T_{max}, the total time of the simulation (a user specified duration). One can think of each logical process as residing on a separate processor, but this is not necessary. In fact, all the logical processes may reside on a single processor. LPs operate independently and communicate with each other only if the physical processes being simulated by the LPs are connected. For example, logical process A (LP_A) connects to LP_B which is in turn connected to LP_C and LP_D etc. Every path which can be traversed by a customer in the physical system must correspond to a logical communication path in the simulation system. Messages passed between LPs in our queueing example are the actual customers flowing through the system.

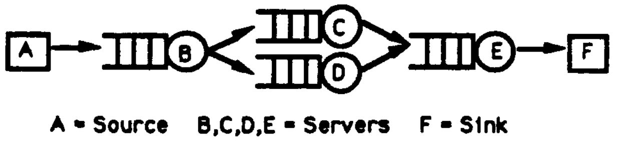


Figure 1: Example queueing network

Each logical process could be placed on its own processor, and one might hope that we could then gain speedup proportional to the number of processors used. Unfortunately, this is often not the case as the system being simulated may have only limited parallelism. Also, the PDES algorithms themselves limit parallelism in their attempt to prevent the simulation from deadlocking and to ensure correctness.

*This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory.

Several competing techniques have been developed to address deadlocking and correctness. One [Peacock et al. 1979] is described as a synchronous approach which keeps logical process clocks in synchronization while another [Jefferson 1985] is an asynchronous strategy which uses a rollback mechanism which is invoked only when needed for synchronization.

2.1 Time-Stepped Simulation

Distributed time-stepped simulation [Peacock et al. 1979] is accomplished by keeping all the local clocks in strict synchronization. At any point in real time each LP's local clock will have the same value as any other LP's clock. As the simulation runs, the local clocks take on a sequence of discrete values $(t_0, t_1, t_2, \ldots)$ each differing by an amount Δ. All processors must complete execution of events up to t_i before any processor begins processing at t_{i+1}. Each processor may have a different amount of work to do at each time step or some may operate at different speeds so that many processors may have to wait for the slowest one to complete execution of the i^{th} step. Time-stepped simulation is attractive due to its simplicity of implementation. By keeping all the LPs processing at the same simulation time, deadlocks cannot occur and no further effort needs to be expended in guaranteeing the correctness of the simulation. Time-stepped simulation is an example of the synchronous approach.

2.2 Time Warp

Our asynchronous example comes from one of the more recent developments in the area of PDES; the so-called optimistic strategies. One such strategy is called Time Warp and was developed by Jefferson [Jefferson 1985]. The basic idea is that the requirement of keeping each LP in strict synchronization (keeping local clocks the same), even when it isn't necessary, may lead to a degradation in performance. The Time Warp mechanism allows LPs to race forward as quickly as possible. If a message arrives which has a lower timestamp than the value of the LP's clock, indicating the LP proceeded with incomplete information, the LP is "rolled back" to the time of the incoming message. This can be accomplished because the system periodically saves the state of the LP. Any effects of having advanced too far (i.e. erroneous messages) are canceled through an elegant technique using anti-messages [Jefferson 1985].

3 The Models

We have opted to use very simple models of the two approaches in order to assess the potential improvement of the asynchronous versus the synchronous strategy. Our model of the time-stepped (synchronous) strategy will provide us with an accurate estimate of its time to complete a simulation, while the model for the asynchronous strategy will provide us with an overly low estimate of its expected completion time. Therefore, we establish an upper bound on the potential improvement of the asynchronous strategy over the time-stepped method.

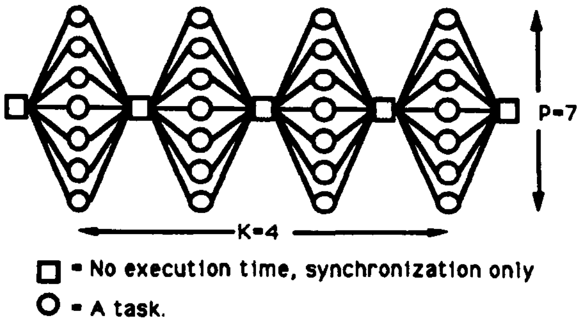


Figure 2: Synchronous Task Graph

To analyze the two techniques, we propose the following model: P processors each execute K tasks (events) sequentially. Each processor p must perform tasks $T_{p1} \cdots T_{pk} \cdots T_{pK}$ in sequential order. K determines the "size" of the simulation. A task will take a random amount of time to complete execution on any processor.

Our model of the synchronous approach is based on the idea that an LP must wait for all other LPs to complete a step before continuing. Each processor must wait until *every* processor has completed its i^{th} task prior to beginning execution of the $(i+1)^{st}$ task. This is essentially a "staged" execution with K stages where each stage takes as long as the slowest processor. This task graph is shown in Figure 2 for $K = 4$ and $P = 7$.

The asynchronous strategy has no such "staging" restriction, and, moreover, in the best possible circumstance no rollbacks will occur. We allow each processor to execute its tasks in order as fast as it can, without waiting for the other processors to finish. The total time to finish is simply the time that the slowest processor takes to complete its K tasks. To keep the model simple we are assuming no rollbacks; it is as if each processor never has to wait for any messages from other processors, and that all messages arrive in timestamp order. The asynchronous task graph is shown in Figure 3.

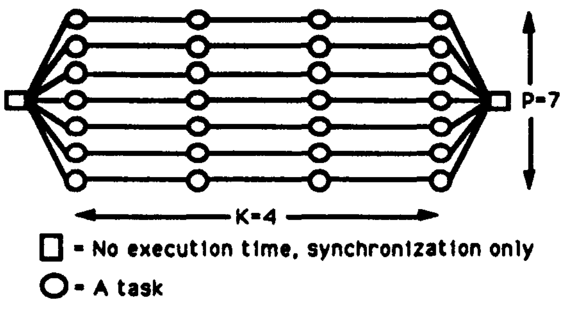


Figure 3: Asynchronous Task Graph

4 Space of Synchronization Methods

Though the models we are using are extremely simple, we believe they provide us with very important information. Our time-stepped model (Figure 2) requires the most synchronization, and therefore will take the longest time to complete execution of any system which exhibits full parallelism in each stage. Our asynchronous model (Figure 3) shows the least amount of internal synchronization (none) and should complete execution in less time than any other method. Therefore, we believe that these two models span the range of possibilities and give a good indication of the maximum performance improvement that could be gained by using the asynchronous strategy.

5 Exponentially Distributed Task Times

If we model each task execution time with an exponential distribution and treat the processors as identical, the expected time for the synchronous strategy is K times the maximum of P exponentials, while the expected time for the asynchronous strategy is the maximum of P K-stage Erlangs. We now proceed to calculate the ratio R_e of the expected completion times for exponentially distributed task times.

5.1 Time-Stepped (Synchronous) Model

Let $T =$ the maximum of P exponentials with mean $\frac{1}{\mu}$. The cumulative distribution (PDF) of T is

$$F_T(x) = (1 - e^{-\mu x})^P, \qquad (1)$$

with density function

$$f_T(x) = P(1 - e^{-\mu x})^{P-1} \mu e^{-\mu x}. \qquad (2)$$

Using the PDF we can calculate the expected value of T [Kleinrock 1975].

$$\begin{aligned} E[T] &= \int_0^\infty (1 - F_T(x))dx \\ &= \int_0^\infty \left[1 - (1 - e^{-\mu x})^P\right] dx \\ &= \int_0^\infty \left[1 - \sum_{i=0}^{P} \binom{P}{i} 1^{P-i}(-e^{-\mu x})^i\right] dx \\ &= \sum_{i=1}^{P} \binom{P}{i} (-1)^{i+1} \int_0^\infty (e^{-\mu i x})dx \end{aligned}$$

Since [Graham et al. 1989]

$$\sum_{i=1}^{P} \binom{P}{i} \frac{1}{i}(-1)^{i+1} = \sum_{i=1}^{P} \frac{1}{i},$$

$$E[T] = \frac{1}{\mu} \sum_{i=1}^{P} \frac{1}{i}. \qquad (3)$$

We now define T_s as the time for all K stages to complete. Clearly, $E[T_s] = KE[T]$. So the final equation for $E[T_s]$ where P is the number of processors and K is the number of steps is:

$$E[T_s] = \frac{K}{\mu} \sum_{i=1}^{P} \frac{1}{i}. \qquad (4)$$

An excellent approximation for this is [Jolley 1961]:

$$E[T_s] \approx \frac{K}{\mu}\left(E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}\right). \qquad (5)$$

where $E =$ Euler's Constant ≈ 0.57722.

5.2 Time Warp (Asynchronous) Model

We define T_a as the maximum of P K-stage Erlangs where each stage has mean $\frac{1}{\mu}$. The probability density function of a single K-stage Erlang is

$$f_t(x) = \frac{\mu e^{-\mu x}(\mu x)^{K-1}}{(K-1)!}. \qquad (6)$$

The PDF can be found either by direct integration of the density function, or by realizing that the probability that a K-stage Erlang takes time less than or equal to t is one minus the probability that it takes time greater than t, which is simply one minus the probability that there are less than K arrivals in the interval $[0 - t]$ from a Poisson process at rate μ. Therefore

$$F_t(x) = \int_0^x f_t(x)dx = 1 - e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}. \qquad (7)$$

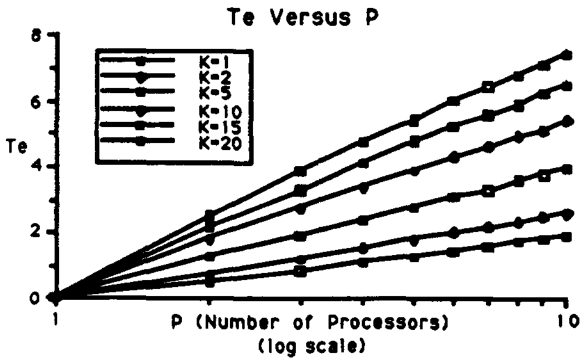


Figure 4: T_e versus $\ln P$.

The cumulative distribution of the maximum of P K-stage Erlangs is:

$$F_{T_a}(x) = \left(1 - e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}\right)^P. \quad (8)$$

Using $F_{T_a}(x)$ we can calculate the expectation of T_a.

$$E[T_a] = \int_0^\infty [1 - F_{T_a}(x)]\, dx$$

Thus,

$$E[T_a] = \int_0^\infty 1 - \left(1 - e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}\right)^P dx$$

$$= \int_0^\infty \sum_{j=1}^{P} \binom{P}{j} (-1)^{j+1} \left(e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}\right)^j dx$$

Unfortunately, this equation has no closed form expression for the integral. By decomposing $E[T_a]$ into two components: the mean of a K-stage Erlang and $T_e \triangleq$ the difference between the mean and the expected value of the max of P K-stage Erlangs, we can approximate $E[T_a]$.

$$\begin{aligned} E[T_a] &= \text{Mean of } K\text{-stage Erlang} + T_e \\ &= \frac{K}{\mu} + T_e \end{aligned}$$

An excellent approximation for T_e when $K > 1$ and $P > 1$ is

$$T_e \approx \frac{1}{\mu}\left((C \ln^2 K + D) \ln P + AK + B\right). \quad (9)$$

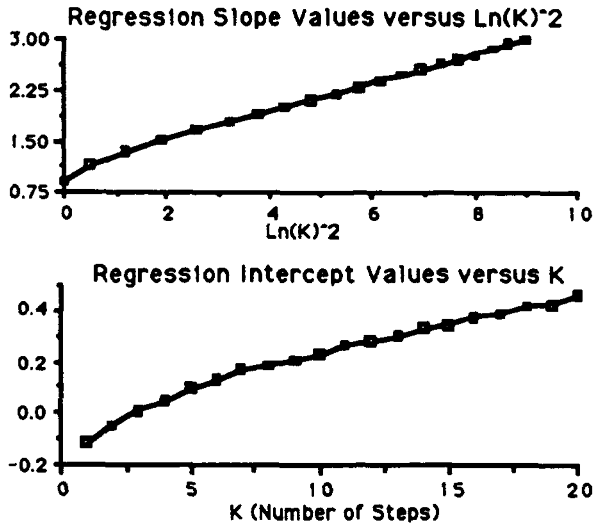


Figure 5: Regression Values

Where

$$A = 0.02244 \approx 0.02 \quad B = 1.14571 \approx 1.15$$
$$C = 0.22278 \approx 0.22 \quad D = 0.55957 \approx 0.56.$$

This approximation was developed by using least squares regression techniques three times. It was first noticed that for a fixed K, T_e seemed to be directly related to $\ln P$. This is clearly seen in Figure 4. For each value of $k \leq K$ we performed a linear regression for T_e such that $(T_e)_k = m_k(\ln P) + b_k$, thus generating K slopes and intercepts, one set for each value of k. Then, it appeared that the slopes for each k approximation were linearly related to $\ln^2 k$, while the intercepts seemed linearly related to k. This can be seen in Figure 5. Therefore, a second regression was performed on the slope values versus $\ln^2 k$ (generates the values for the constants C and D), while a third regression was performed on the intercept values versus k (generates the values for the constants A and B). Figure 6 shows the approximation compared to simulation for values of K and P between one and ten and Figure 7 shows the comparison for $K, P >= 100$.

5.3 Relative Performance

Let us look at the ratio, R_s of the two approximations.

$$E[T_s] \approx \frac{K}{\mu}\left(E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}\right)$$

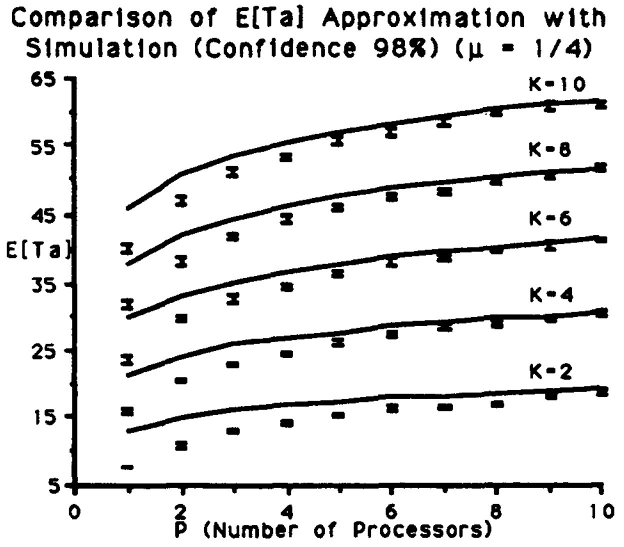


Figure 6: Comparison of Approximation and Simulation for $K \leq 10$.

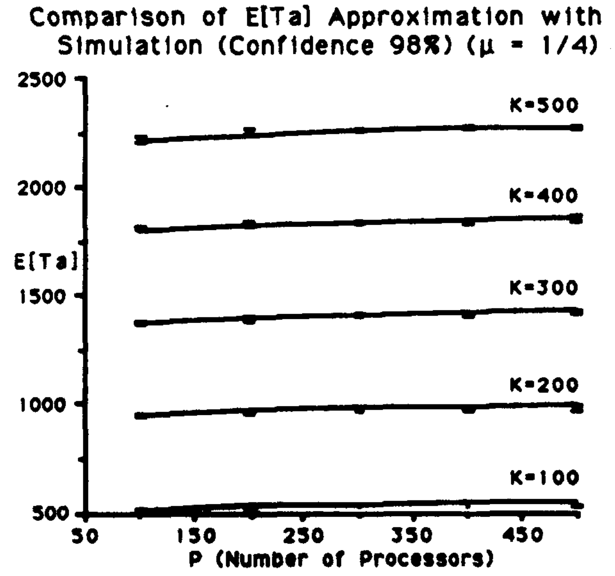


Figure 7: Comparison of Approximation and Simulation for $K \geq 100$.

$$E[T_a] \approx \frac{K}{\mu} + \frac{1}{\mu}\left((C\ln^2 K + D)\ln P + AK + B\right)$$

$$\begin{aligned} R_e &= \frac{E[T_s]}{E[T_a]} \\ &= \frac{\frac{K}{\mu}\left(E + \ln(P) + \frac{1}{2P} - \frac{1/12}{P(P+1)}\right)}{\frac{1}{\mu}\left((C\ln^2(K) + D)\ln(P) + (1+A)K + B\right)} \\ &= \frac{E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}}{\frac{(C\ln^2 K + D)\ln P}{K} + (1+A) + \frac{B}{K}} \end{aligned}$$

Taking the limit as the size of the simulation increases ($K \to \infty$) and assuming that $K >> \ln P$ we get

$$\lim_{K\to\infty} R_e = \frac{E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}}{(1+A)}.$$

Finally, for large P

$$\lim_{K\to\infty, P\to\infty} R_e \approx \frac{\ln P}{(1+A)} \approx \frac{\ln P}{1.02} \approx \ln P. \quad (10)$$

Thus, in the limit as the size of the simulation increases to infinity, the asynchronous approach could, at most, complete $\ln P$ times as fast as the time-stepped method on average. We can derive this result by appealing to intuition. We have exponential task times where each task takes, on average, $\frac{1}{\mu}$ seconds to complete. For synchronized execution, basic principles (Section 5.1) tell us that each stage will take time proportional to $\frac{1}{\mu}\ln P$ on the average for a total expected time of $\frac{K}{\mu}\ln P$. The asynchronous execution on average takes time equal to $\frac{K}{\mu}$ plus T_e a term which is small compared to $\frac{K}{\mu}$ for large K. Therefore, the ratio of the two times should be $\ln P$. It should be noted that a trivial lower bound on T_a (thus an upper bound on R_e) is found by simply using the mean of P K-stage Erlangs. Our approximation (Equation 9) confirms this result since A,B,C and D are all non-negative.

Additionally, if we believe that no method could achieve a speedup greater than P for P processors over execution on a single processor, then any time-stepped method is limited to a maximum speedup of $\frac{P}{\ln P}$. These results depend on the assumption of an exponential distribution for task times. The next section uses a uniform [0-X] distribution for task execution times.

6 Uniformly Distributed Task Times

If we make the assumption that the task times are uniformly distributed between 0 and X, we calculate a different limiting value for the ratio of completion times. It is easy to show that the maximum of P uniformly distributed random variables is $X\frac{P}{P+1}$. We immediately find that

$$E[T_s] = KX\frac{P}{P+1}. \quad (11)$$

Fortuitously, we can use the same regression technique used with the exponential distribution in Section 5.2 to develop an accurate approximation for $E[T_a]$. Therefore

$$E[T_a] = \frac{KX}{2} + T_e$$

$$\approx \frac{KX}{2} + X\left((C\ln^2 K + D)\ln P + AK + B\right). \quad (12)$$

Where

$$A = 0.012384 \approx 0.01 \quad B = 0.330691 \approx 0.33$$
$$C = 0.053147 \approx 0.05 \quad D = 0.125102 \approx 0.13.$$

Finally, we look at the ratio of the expected completion times.

$$R_u = \frac{E[T_s]}{E[T_a]}$$

$$= \frac{XK\frac{P}{P+1}}{X\left((C\ln^2(K) + D)\ln(P) + (A + 1/2)K + B\right)}$$

$$= \frac{\frac{P}{P+1}}{\frac{(C\ln^2 K+D)\ln P}{K} + (A + 1/2) + \frac{B}{K}}$$

Taking the limit as the size of the simulation increases ($K \rightarrow \infty$) and assuming that $K >> \ln P$ we get

$$\lim_{K\rightarrow\infty} R_u = \frac{P}{(A + 1/2)(P + 1)}.$$

Finally, for large P

$$\lim_{K\rightarrow\infty, P\rightarrow\infty} R_u = \frac{1}{(A + 1/2)} \approx \frac{1}{0.51} \approx 2. \quad (13)$$

Thus, when using a uniform distribution, the asynchronous strategy is only able to complete in roughly half the time, regardless of the number of processors used (as compared to our previous case where the task execution times had exponential tails and the asynchronous strategy was able to gain its $\ln P$ performance improvement). This result should apply to any distribution with finite support since the maximum of many such random variables will invariably approach the upper limit.

Again, we can appeal to intuition to find the speedup ratio. For large P the synchronized execution will take X seconds per stage (the max) on average for a total time of KX. The asynchronous system, on average, will take time equal to $\frac{KX}{2} + T_e$. Since T_e is small compared to $\frac{KX}{2}$ for large K, the speedup ratio should be 2.

As before, if we assume that no method can achieve speedup greater than P over a sequential execution, then the synchronous strategy could possibly have speedup proportional to P when the task times are uniformly distributed.

7 Conclusions

We have shown that an asynchronous distributed simulation strategy can have at most a $\ln P$ performance improvement over a time-stepped method, in the case where task times are exponentials. We conjecture that this result is due to the infinite tail on the exponential distribution and may therefore be applicable to distributions with exponential tails. The improvement when using a distribution with finite support (e.g. uniform) is reduced to a constant amount independent of P.

References

[1] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Co., 1989.

[2] David R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3), July 1985.

[3] L.B.W. Jolley. *Summation of Series*. Dover Publications, Inc., second revised edition, 1961.

[4] Leonard Kleinrock. *Queueing Systems: Volume 1: Theory*. John Wiley and Sons, Inc., 1975.

[5] Jayadev Misra. Distributed discrete-event simulation. *Computing Surveys*, 18(1), March 1986.

[6] J. Kent Peacock, J.W. Wong, and Eric G. Manning. Distributed simulation using a network of processors. *Computer Networks*, 3(1):44–56, 1979.

Bob Felderman was born in Chicago, Illinois in 1962. He graduated Magna Cum Laude from Princeton University in 1984 with a double major in Electrical Engineering & Computer Science and Systems Engineering. After spending a year at Hughes Aircraft Company working on guidance systems for torpedos, he returned to the good life of academia, completed his Master's degree in Computer Science at UCLA in 1986 and is currently pursuing a Ph.D. in Computer Science specializing in distributed systems. In his spare time he can be found in the great outdoors usually with frisbee in hand.

On Distributed Systems Performance *

Leonard KLEINROCK
Computer Science Department, University of California, Los Angeles, CA 90024-1596, USA

Abstract. The behavior of two interacting processes in a distributed processing environment is analyzed. This problem represents a class of problems which we will confront in the next decade as distributed systems are implemented.

Keywords. Distributed systems, distributed simulation, timewarp, synchronization, speedup, parallel processing, rollback, communicating processes.

Leonard Kleinrock is a Professor of Computer Science at U.C.L.A. He received the B.S. degree in electrical engineering from the City College of New York in 1957 (evening session) and the M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institute of Technology in 1959 and 1963, respectively.

He joined the Faculty at the University of California, Los Angeles, in 1963. His research interests focus on local area networks, computer networks and performance evaluation of distributed systems. He has had over 150 papers published and is the author of five books. He is principal investigator for the DARPA Parallel Systems Laboratory contract at U.C.L.A. He is also founder and CEO of Technology Transfer Institute, a computer-communications seminar and consulting organization located in Santa Monica, CA.

Dr. Kleinrock is a member of the National Academy of Engineering, a Guggenheim Fellow, and a member of the Computer Science and Technology Board of the National Research Council. He has received numerous best paper and teaching awards, including the ICC 1978 Prize Winning Paper Award, the 1976 Lanchester Prize for outstanding work in Operations Research, and the Communications Society 1975 Leonard G. Abraham Prize Paper Award. In 1982, as well as having been selected to receive the C.C.N.Y. Townsend Harris Medal, he was co-winner of the L.M. Ericsson Prize, presented by His Majesty King Carl Gustaf of Sweden, for his outstanding contribution in packet switching technology. In 1986, he received the 12th Marconi International Fellowship Award, presented by His Royal Highness Prince Albert, brother of King Baudoin of Belgium, for his pioneering work in the field of computer networks.

* This research was supported by DARPA grant MDA 903-87-C-0663.

North-Holland
Computer Networks and ISDN Systems 20 (1990) 209–215

1. Introduction

The design and performance evaluation of distributed systems is an important and difficult problem and one which will occupy our attention for the next decade. Indeed it represents an example of the class of resource allocation problems with which we have been wrestling for many years in a variety of different contexts. For example, the problem of designing the operating system for time-shared computers was a major issue in the 1960's, the issue of wide-area network design and access occupied our interest in the 1970's, the problem of local area network design was our focus for the decade of the 1980's and we foresee that the general problem of distributed processing will surely occupy our attention for the coming decade of the 1990's.

One aspect of the problem has to do with resolving conflicts. This issue manifests itself both in centralized, as well as in distributed systems. The problem arises when more than one user requires access to the same resource at the same time. Usually we cannot predict exactly *when* a user will require access to the resource, we cannot predict *how long* each user will hold the resource once he gains access, most users only require the *occasional use* of the resource, and, in addition, when a user asks for access he usually expects *immediate access* to that resource. This presents a nasty set of requirements on the part of the user and we refer to such a class of users as being *bursty and asynchronous*. There are four canonical ways of resolving conflicts. The first is *queueing*: here one user gets access to the resource while the others wait for their turn. The second resolution method is that of *splitting*: here, the resource is split into as many pieces as there are competing users and each user gets a piece of the resource. The third canonical resolution method is *blocking*: here, one user gets access to the resource and the others are asked to go away. The fourth method is *smashing*: here, if more than one user asks for access, no one is given access. Examples of each of these systems are prevalent throughout the computer and communication industry. Of course, one may use hybrid mixtures of these four canonical resolution methods.

Queueing is perhaps the most common conflict resolution method and is often found in our current technology. We are all familiar with the price one pays for queueing, namely, an increase in response time due to the sharing of a resource with other users (see for example, [1,2]). However, in a distributed system, we are confronted with additional access problems and delays beyond those due to pure queueing. This comes about for many reasons discussed in the next section, all related to the fact that one cannot form a queue for free in a distributed environment. In particular, we will focus later in this paper on the specific issue of *synchronization* among coupled processes.

2. Problems of Distributed Access

Once we distribute resources and users, we are faced with a number of difficult conflict resolution and access problems. One way to divide these problems into recognizable systems is to consider a two-dimensional description where the first dimension describes the degree of coupling among the distributed processes (i.e. the amount of communication and interaction among them) and where the second dimension describes the distance that separates these processes. This may be seen in Fig. 1.

In this figure, the items above the dashed line within a quadrant refer to *processing* applications, whereas items below the dashed line refer to *communications* applications. Applications in the case of tightly coupled processes which are close to each other include parallel processing (a number of processors cooperating in the execution of a single problem) as well as back plane buses and high speed LANs. In the case of loosely coupled processes which are close to each other, we include such things as distributed processing (a number of processors, each possibly working on a different problem with occasional interaction among these processors) as well as LANs and MANs. Loosely coupled processes which are separated by large distances include applications such as distributed access (e.g. remote access to a data base) as well as wide-area networks. However, in the case of tightly coupled processes which are far from each other, we find very few applications, all of which are extremely difficult due to the large amount of interaction required at what may be long distances and/or long delays.

		COUPLING	
		TIGHT	LOOSE
DISTANCE	CLOSE	PARALLEL PROCESSING ------------------------ BACK PLANE BUS HIGH SPEED LANs	DISTRIBUTED PROCESSING ------------------------ LANs MANs
	FAR	HA !	DISTRIBUTED ACCESS ------------------------ WANs

Fig. 1. Examples of distributed systems.

A major problem that we face in distributed access is that we usually lack *global knowledge* regarding the system state. A number of things contribute to this lack of global knowledge and manifest themselves as problems. For example, a long distance between users which must interact becomes a problem with regard to the speed with which they can interact and the bandwidth of the communications of that interaction (for example, it takes light approximately 15 000 microseconds to cross the United States!); thus, the time for state information to be exchanged between processes may be seriously delayed leading to a lack of global knowledge. Another problem is that not all processes (users) may be in immediate communication with each other; for example, two users may not be able to hear each other and may require intermediate users to relay information between them. Sometimes the state information we get is incomplete and sometimes it is incorrect. Even if it is complete and correct, it may be that the state information is stale by the time we have an opportunity to use it, since it may take a user a certain amount of time to get to the location where he can use the information he gathered previously. Thus, lack of global knowledge prevents perfect use of all system resources in addition to any congestion problems which would arise in a centralized configuration with perfect knowledge.

Another source of difficulty in distributed systems has to do with access to resources themselves. For example, even though there is a set of resources in a system, not all users may be allowed access to certain of the resources. Livny and Mel-

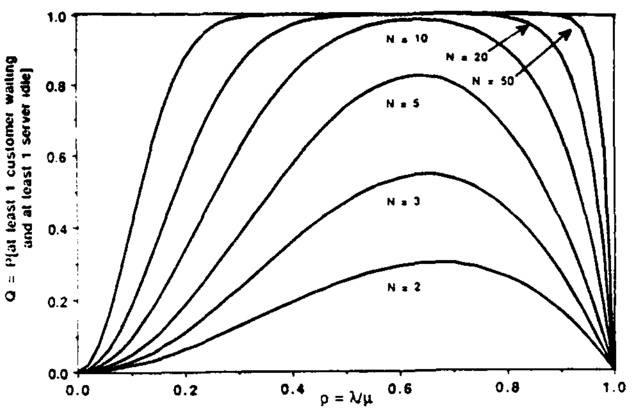


Fig. 2. Performance of N M/M/1 queueing systems.

man [3], studied the case of N independent M/M/1 queueing systems, each with its own independent arrival process at rate λ and each with its own server at rate μ. For this system, the probability Q that at least one customer is waiting in some queue and at least one server is idle is given by

$$Q = 1 - \rho^N + (1-\rho)^N\left[\rho^N - (1+\rho)^N\right]. \qquad (2.1)$$

In Fig. 2, we plot Q as a function of $\rho = \lambda/\mu$ with N as a parameter. From this result we see that Q approaches 1 for all values of ρ in the range $0 < \rho < 1$, as the number of independent systems increases ($N \to \infty$) indicating serious degradation to system performance. Of course, this is only a simple measure of system degradation but does indicate the cost of prohibited access.

One way to improve performance of these N independent queueing systems is to allow some jockeying among the queues. A model which introduces this jockeying is to allow each user from queue n ($n = 1, 2, \ldots, N$) to move to queue $n+1$ (mod N) at a rate α. Clearly, for $\alpha = 0$, we have Livny's model and for $\alpha = \infty$ we have a simple M/M/N queue which provides, in some sense, perfect sharing of the N servers. Unfortunately, this model with jockeying is an unsolved coupled queueing problem (even for the case $N = 2$); coupled queueing problems are very difficult in general.

Another problem with access has to do with the fact that the distributed system may be lossy; by this we mean it is possible, that, as the load increases, the throughput might decrease due to internal waste of resources (see for example [4]).

The access problem may also manifest itself, in some cases, by requiring that resources be used in series (one after the other) rather than all at once. This leads to increased delays and possibly inefficient use of resources. An example here might be that of a message passing through a wide-area network which must hop from channel to channel as it makes its way through the network.

Another manifestation of this access problem is that there may be synchronization and/or precedence constraints in the way in which the tasks required by a user are executed. This arises with the parallel use of processors and possible restrictions on how much parallelism the algorithm or the application allows. It is on this type of problem that we focus in the remainder of this paper. Below, we give a description of the problem, a model of the system, analysis of its behavior and a discussion of the results.

3. A Model for Synchronization Beween Two Processes

Assume we have a job which is partitioned into two processes, each of which is executed on a separate processor. As these processes are executed, we consider that they advance along the x-axis in steps of length one (i.e. they visit the non-negative integers), each beginning at $x = 0$ at time $t = 0$. Each process independently takes an exponentially distributed amount of time, with parameter λ_i ($i = 1, 2$), to advance from position k to position $k+1$ ($k = 0, 1, 2, \ldots$). When process i advances one unit along this axis, it will send a message to the other process with probability q_i ($0 < q_i \leqslant 1$). Upon receiving a message from the other (sending) process, this (receiving) process will do the following:

(1) If its position along the x-axis is equal to or behind the sending process, it will ignore the message.

(2) If it is ahead of the sending process, it will immediately move back (i.e., "rollback") along the x-axis to the current position of the sending process.

This is a simple model of distributed simulation (motivated by the time warp distributed simulation algorithm [5]) where two processors are both working on a simulation job in an effort to speed it up. They both proceed independently until such time as one (slower) process transmits a message in the "past" of the other (faster) process. This causes the faster process to "rollback" to the point that the slower process is at, after which they advance independently again until the next rollback, etc.

Let $F(t)$ = position of the first process (process 1) at time t and let $S(t)$ = position of the second process (process 2) at time t. Further, let

$$D(t) = F(t) - S(t);$$

$D(t) = 0$ whenever (2) occurs (i.e., a rollback). We are interested in studying the Markov process $D(t)$. From our assumptions that $F(0) = S(0) = 0$, we have $D(0) = 0$. Clearly, $D(t)$ can take on any integer value (i.e., it certainly can go negative). We will solve for

$$p_k = \lim_{t\to\infty} P[D(t) = k], \quad k = \dots, -2, -1, 0, 1, 2, \dots \tag{3.1}$$

namely, the equilibrium probability for the Markov Chain $D(t)$. Moreover, we will find the average separation between processes as well as the speedup with which the computation proceeds when using two processors relative to the use of a single processor as described below.

Our model is that of a discrete state, continuous time process. Some previous work on variations of this model already exists. For example, Mitra and Mitrani [6] studied a related model in which they considered a continuous state, discrete time model not unlike the one we have described here. Their results are similar to ours in some ways although their method of analysis appears to be far more complex than ours. Lavenberg, Muntz, and Samadi [7] provide an approximate analysis of a continuous time, continuous state model. (We have also completed the analysis of the discrete time, discrete state case which will be published elsewher.) Moreover, Felderman and Kleinrock [8] give an upper bound on the gain in speedup that P unsynchronized processors can achieve relative to P processors which are forced to synchronize at every step.

4. Analysis

Let us analyze the behavior of these two coupled processes which we have modeled as a one-dimensional discrete state, continuous time Markov Chain. The following balance equations are easily obtained:

$$(\lambda_1 + \lambda_2)p_k = \lambda_1 p_{k-1} + \lambda_2 \bar{q}_2 p_{k+1}, \quad k = 1, 2, \dots, \tag{4.1}$$

$$(\lambda_1 + \lambda_2)p_{-k} = \lambda_2 p_{-(k-1)} + \lambda_1 \bar{q}_1 p_{-(k+1)}, \quad k = 1, 2, \dots, \tag{4.2}$$

$$(\lambda_1 + \lambda_2)p_0 = \lambda_2 q_2 \sum_{k=1}^{\infty} p_k + \lambda_1 q_1 \sum_{k=1}^{\infty} p_{-k} + \lambda_2 \bar{q}_2 p_1 + \lambda_1 \bar{q}_1 p_{-1} \tag{4.3}$$

where $\bar{q}_i = 1 - q_i$.

We solve this system of linear difference equations using the usual approach of z-transforms [1] by defining the two transforms

$$R(z) = \sum_{k=1}^{\infty} p_k z^k, \tag{4.4}$$

$$Q(z) = \sum_{k=1}^{\infty} p_{-k} z^k. \tag{4.5}$$

We further define the relative speed parameter

$$a = \frac{\lambda_1}{\lambda_1 + \lambda_2}. \tag{4.6}$$

Multiplying (4.1) by z^k and summing over the range $k = 1, 2, \dots$ we obtain

$$R(z) = \frac{z(\bar{a}\bar{q}_2 p_1 - a p_0 z)}{az^2 - z + \bar{a}\bar{q}_2} \tag{4.7}$$

where $\bar{a} = 1 - a$. Similarly, multiplying (4.2) by z^k and summing over the range $k = 1, 2, \dots$ we obtain

$$Q(z) = \frac{z(a\bar{q}_1 p_{-1} - \bar{a} p_0 z)}{\bar{a}z^2 - z + a\bar{q}_1}. \tag{4.8}$$

Note the duality between $R(z)$ and $Q(z)$ with regard to the variables a, q_1, and q_2.

The denominator roots (i.e. the poles) of $R(z)$ are given by

$$r_1 = \frac{1 - \sqrt{(1-2a)^2 + 4a\bar{a}q_2}}{2a},$$

$$r_2 = \frac{1 + \sqrt{(1-2a)^2 + 4a\bar{a}q_2}}{2a}. \tag{4.9}$$

Similarly, the poles of $Q(z)$ are given by

$$s_1 = \frac{1 - \sqrt{(1-2a)^2 + 4a\bar{a}q_1}}{2\bar{a}},$$

$$s_2 = \frac{1 + \sqrt{(1-2a)^2 + 4a\bar{a}q_1}}{2\bar{a}}. \tag{4.10}$$

It is easy to show that these four poles are real and that $0 \leqslant r_1 \leqslant 1$, $1 \leqslant r_2$, $0 \leqslant s_1 \leqslant 1$ and $1 \leqslant s_2$. Since $R(z)$ and $Q(z)$ are both analytic in the region $|z| \leqslant 1$, it must be that numerator $(R(r_1)) = 0$ and numerator $(Q(s_1)) = 0$. From this observation and from (4.7) and (4.8) we have

$$p_1 = \frac{ar_1}{\bar{a}\bar{q}_2} p_0. \tag{4.11}$$

$$p_{-1} = \frac{\bar{a}s_1}{a\bar{q}_1} p_0. \tag{4.12}$$

In addition, by conserving probability we have

$$p_0 + R(1) + Q(1) = 1. \tag{4.13}$$

From these last three we readily find

$$p_0 = \frac{(r_2 - 1)(s_2 - 1)}{r_2 s_2 - 1}. \tag{4.14}$$

From (4.11) and (4.12) we may simply rewrite $R(z)$ and $Q(z)$ as

$$R(z) = \frac{zp_0}{r_2 - z}, \qquad Q(z) = \frac{zp_0}{s_2 - z}. \tag{4.15}$$

We may now invert both $R(z)$ and $Q(z)$ to give us the equilibrium distribution for our Markov Chain, namely,

$$p_k = \begin{cases} p_0\left(\dfrac{1}{r_2}\right)^k, & k = 0, 1, 2, \ldots, \\ p_0\left(\dfrac{1}{s_2}\right)^{-k}, & k = 0, -1, -2, \ldots. \end{cases} \tag{4.16}$$

Equation (4.16), along with (4.9), (4.10) and (4.14), give us the complete solution to the Markov Chain.

To find the average separation between these two processes on the x-axis, namely, $\bar{K} = \lim_{t \to \infty} E[\,|D(t)|\,]$, we calculate as follows:

$$\bar{K} = \sum_{k=1}^{\infty} k(p_k + p_{-k})$$

which gives us

$$\bar{K} = p_0\left[\frac{r_2}{(r_2 - 1)^2} + \frac{s_2}{(s_2 - 1)^2}\right]. \tag{4.17}$$

Let us now calculate the speedup S which is defined as the rate at which the two processor system carries out useful processing divided by the rate at which an equivalent single processor carries out useful processing. We define the equivalent single processor as one which moves a process along the time axis at a rate equal to the average rate of the two original processes, namely: $\frac{1}{2}(\lambda_1 + \lambda_2)$. In this single processor case, there is no rollback to worry about and so useful progress occurs at the rate $\frac{1}{2}(\lambda_1 + \lambda_2)$. In the two processor case, useful progress is equal to the sum of the two rates minus the expected rollback for each process. If $D(t) = k$ at time t, and if the next advance along the x-axis is made by the lagging process which also causes the leading process to rollback (with probability q_2), then the leading process will be rolled back only a distance $k - 1$ since the lagging process just advanced one step along the x-axis. Thus, we see that the rate at which the two processor system advances, on average, is given by

$$\lambda_1 + \lambda_2 - \sum_{k=1}^{\infty} \lambda_2 q_2 p_k (k-1) - \sum_{k=1}^{\infty} \lambda_1 q_1 p_{-k}(k-1).$$

Thus, the speedup is given by

$$S = 2 - 2\bar{a}q_2 \sum_{k=1}^{\infty} p_k(k-1) - 2aq_1 \sum_{k=1}^{\infty} p_{-k}(k-1).$$

This leads us to the following general expression for the system speedup:

$$S = 2\left[1 - \frac{\bar{a}q_2 p_0}{(r_2 - 1)^2} - \frac{aq_1 p_0}{(s_2 - 1)^2}\right]. \tag{4.18}$$

5. The Symmetric Case $q_1 = q_2$

In this section we consider the symmetric case where $q_1 = q_2 = q$; that is, each process has the same probability of sending a message to the other process. We now have $ar_1 = \bar{a}s_1$ and $ar_2 = \bar{a}s_2$.

The speedup is shown in Fig. 3 as a function of a and q. For $a = \frac{1}{2}$, the speedup rises continously to its maximum value of $S = 2$ as $q \to 0$. For $q = 0$, $S = 2$ for all a but S has a discontinuity for all $a \neq \frac{1}{2}$; this discontinuity is not shown clearly in Fig. 3. (For $q = 0$, no rollbacks occur and it is intuitively clear that $S = 2$.) Note for $q > 0$ that, when $a \to 0$ or $a \to 1$ (that is, $\lambda_1 \to 0$ or $\lambda_2 \to 0$), then the speedup goes to 0; this is the case, since one process moves extremely slowly (compared to the equivalent single process) and it

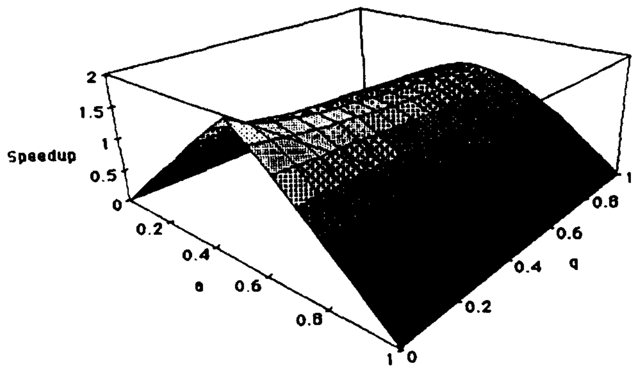


Fig. 3. Speedup for the symmetric case, $q_1 = q_2 = q$.

will occasionally drag the faster process back to its lagging position.

6. The Balanced Case $\lambda_1 = \lambda_2$

Here we consider the balanced case where both processes move at the same rate giving us $a = (1 - a) = \frac{1}{2}$. We see that $r_1 = 1 - \sqrt{q_2}$, $r_2 = 1 + \sqrt{q_2}$, $s_1 = 1 - \sqrt{q_1}$ and $s_2 = 1 + \sqrt{q_1}$. In addition,

$$p_0 = \frac{\sqrt{q_1 q_2}}{\sqrt{q_1 q_2} + \sqrt{q_1} + \sqrt{q_2}}. \tag{6.1}$$

Furthermore, in this case, we find that the speed-up is simply

$$S = \frac{2\left(\sqrt{q_1} + \sqrt{q_2}\right)}{\sqrt{q_1 q_2} + \sqrt{q_1} + \sqrt{q_2}}. \tag{6.2}$$

In Fig. 4 we show the speedup as a function of q_1 and q_2. Note, of course, that the speedup goes to 2 for $q_1 = q_2 = 0$ and goes to $\frac{4}{3}$ for $q_1 = q_2 = 1$.

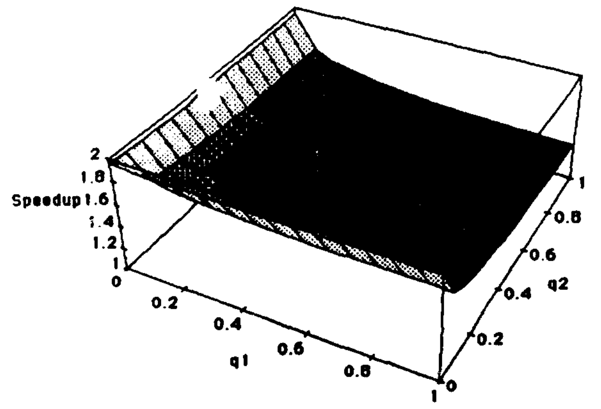


Fig. 4. Speedup for the balanced case, $\lambda_1 = \lambda_2$ $(a = \frac{1}{2})$.

7. The Symmetric Balanced Case: $q_1 = q_2$ and $\lambda_1 = \lambda_2$

In this symmetric balanced case, where both processes move at the same speed and both have the same probability, $q_1 = q_2 = q$, of passing a message to the other process, we obtain great simplifications. In particular, we have $r_1 = s_1 = 1 - \sqrt{q}$ and $r_2 = s_2 = 1 + \sqrt{q}$. Note, again, that $a = (1 - a) = \frac{1}{2}$. In this case we find,

$$p_0 = \frac{\sqrt{q}}{2 + \sqrt{q}}. \tag{7.1}$$

The average process separation becomes

$$\bar{K} = \frac{2(1 + \sqrt{q})}{\sqrt{q}(2 + \sqrt{q})}. \tag{7.2}$$

The speedup is given by

$$S = \frac{4}{2 + \sqrt{q}}. \tag{7.3}$$

The speedup function in this very special case is shown in Fig. 5. Note for $q = 0$ that $S = 2$ whereas for $q = 1$ we have $S = \frac{4}{3}$. We can see this last result intuitively as follows. If each process always sends a message to the other process when it advances, then the time for both processes to advance one unit is equal to the maximum of two exponential delays which we know is equal to 1.5 times the mean. Thus, the rate of progress for each process is simply $\frac{2}{3}$ times the rate of a single process. Since both are moving at a rate $\frac{2}{3}$, the sum is equal to $\frac{4}{3}$ which yields the result for $q = 1$.

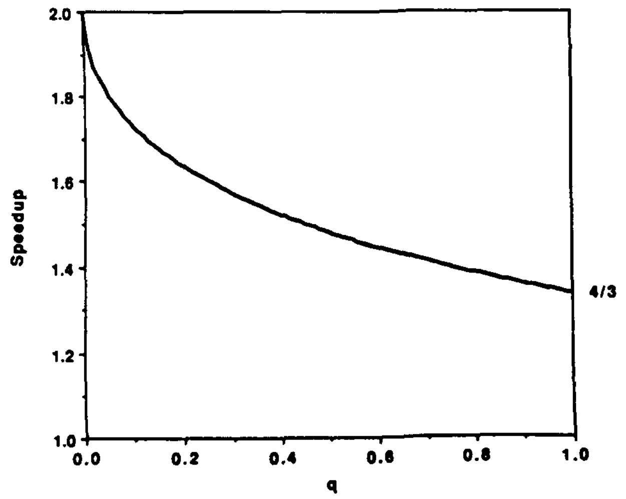


Fig. 5. Speedup for the symmetric, balanced case, $q_1 = q_1 = q$, $\lambda_1 = \lambda_2$ $(a = \frac{1}{2})$.

8. Conclusions

In this paper we have focused on the speedup available when two processes limit each others' rate of progress by passing messages between each other (these messages may cause a process to roll back and therefore waste some of the work it has accomplished). We gave the complete solution for the important system performance variables in the general case. We found for the symmetric balanced case that when $q = 0$ the speedup reaches its maximum value of 2 but that the speedup falls off infinitely quickly as q increases from 0, the limiting speedup being $\frac{4}{3}$ at $q = 1$.

The analysis given here is for only two processors. For larger numbers of interacting processes, the mathematics becomes far more complex and the most likely fruitful approach to understand the interaction among many asynchronous processes would be via approximate solutions.

Synchronization is only one source of overhead one finds in distributed systems. There are numerous other issues which impact the performance of distributed systems. Indeed, we are only beginning to formulate the models and gain the understanding required to analyze, design, and properly implement distributed systems in this coming decade.

References

[1] L. Kleinrock, *Queueing Systems, Vol. I: Theory* (Wiley Interscience, New York, 1975).

[2] L. Kleinrock, *Queueing Systems, Vol. II: Computer Applications* (Wiley Interscience, New York, 1976).

[3] M. Livny and M. Melman, Load balancing in homogeneous broadcast distributed systems, in: *Proc. ACM Computer Network Performance Symposium* (1982) 47–55.

[4] L. Kleinrock, On flow control in computer networks, in: *Proc. Internat. Conf. Communications, Vol. II*, Toronto, Ontario (1978) 27.2.1–27.2.5.

[5] D.R. Jefferson, Virtual time, *ACM Trans. Programm. Languages Systems* 7 (3) (1985) 404–425.

[6] D. Mitra and I. Mitrani, Analysis and optimum performance of two message-passing parallel processors synchronized by rollback, in: *Performance '84* (North-Holland, Amsterdam, 1984) 35–50.

[7] S. Lavenberg, R. Muntz and B. Samadi, Performance analysis of a rollback method for distributed simulation, in: *Performance '83* (North-Holland, Amsterdam, 1983) 117–132.

[8] R. Felderman and L. Kleinrock, An upper bound on the improvement of asynchronous versus synchronous distributed processing in: *Distributed Simulation 1990* (Society for Computer Simulation, 1990) 131–136.

ISDN—The Path to Broadband Networks

LEONARD KLEINROCK, FELLOW, IEEE

Invited Paper

We are in the midst of revolutionary improvements in data communications. The need for connectivity has never been as great as it is today due to the rapid growth of desktop processing machines which must communicate among themselves as well as with centralized computing and database facilities. Alas, in the midst of this progress, we find ourselves burdened by the curse of incompatibility among vendor-specific products, protocols, procedures, and interfaces.

At the same time, the national and international bodies have been hard at work attempting to provide some stability by introducing standards for connectivity. The problem, of course, is one of timing; a premature standard stifles the development of mature technology, while a tardy standard is in danger of being rejected by a community that is locked into irreversible commitments to cumbersome ad hoc solutions. ISDN is an emerging standard which represents an international effort to solve some of our connectivity problems. If it rolls out in a timely fashion and addresses real needs to the end user community, it has a chance for success in the networking world.

The carriers are committed to ISDN and have a clear motivation and potential for succeeding in its development. Narrowband ISDN is a ho-hum service for which some important applications have been identified, but which has not sparked a stampede of acceptance. On the other hand, broadband ISDN (BISDN) is a service that has identified capabilities that are truly exciting and could very well dominate data networking in this decade. The success of BISDN will depend strongly on the rollout of products, the ubiquity of its presence, and the tarriffing of its services.

I. Introduction

Telecommunications is currently a huge industry approaching an annual revenue of $200 000 000 000; it has one of the fastest growth rates of all industries today. Moreover, it is based on some of the most exciting technologies available, changing rapidly, and influencing almost every aspect of business, commerce, education, health, government, and entertainment. Its products are visible to everyone, and yet, the full impact of this juggernaut is not yet appreciated by most observers.

What has caused this enormous growth has been the explosion of digital technology (which itself was fueled by semiconductor electronics, namely, integrated circuits of very large scale, as well as the development of the unbelievable capabilities of fiber-optic communication). This digital technology appeared first as data-processing machines and soon had its impact on data communications. This impact emerged as data communication networks, principally in the form of packet switching in the 1970's [1]. Since then, the data-processing industry and the data communication industry have converged in a fashion that will never again let them separate. You can no longer discuss one without the other.

The product rollout has been staggering and we have been provided a broad range of advanced services, but not without a price. We have now reached a stage of uncontrolled chaos in the marketplace of data processing and data communications. Multivendor systems are almost universal, and the inability of the elements in this heterogeneous environment to interwork is legion. There have been international efforts to bring some order to this chaos through the introduction of standards. Such efforts are almost always slow, laborious, political, petty, boring, ponderous, thankless, and of the utmost criticality. The International Standards Organization has developed the seven-layer Open Systems Interconnection (OSI) reference model for communications. The IEEE 802.X series of standards for communications is growing. We have seen the Consultative Committee for International Telephony and Telegraphy (CCITT) recommendations for their X series of standards proliferate. Moreover, and of most interest to this paper, CCITT has been developing the Integrated Services Digital Network (ISDN) standard since the mid-1970's. The definition and details of this standard are covered elsewhere in these Proceedings.

It is the purpose of this paper to evaluate the effect of ISDN on the field of data networks, to anticipate future directions for this technology, and to discuss how the user should view these developments.

Whereas this paper discusses such issues, the fact is that the underlying issue is really one of *infrastructure*, rather than of ISDN networking by itself. Network technology provides us the capability to install a powerful communications and information technology infrastructure that will enable untold growth and access in the years to come. ISDN is one cornerstone of that technology.

II. Current Status

There are more than 200 000 ISDN access lines installed today, and that number will likely grow to three-quarters of a billion by 1995 [2]. Its use in public networks is clear, and it is beginning to penetrate the private network market as well. It has taken 29 years from the first digital T1 system to today's ISDN developments. 1988 was a critical year, for it was in that year that Signalling System Seven (SS7) installations increased enormously, providing the out-of-band common channel signaling capability on which ISDN is based [3]. We have seen a very rapid rollout from the availability of the basic rate interface (BRI) at 144 kb/s

Manuscript received April 30, 1990; revised September 17, 1990. This work was sponsored by the Defense Advanced Research Projects Agency of the U.S. Department of Defense under Contract MDA 903-87-C-0663.

The author is with t'e Computer Science Department, University of California, Los Angeles, CA 90024-1956.

IEEE Log Number 9041451.

0018-9219/91/0200-0112$01.00 © 1991 IEEE

(2 B + D) and the primary rate interface (PRI) at 23 × 64 kb/s to today's beginning of BISDN at 155 Mb/s and growing to over 13 Gb/s speeds. Indeed, we have already seen the early demonstrations of the 802.6 Metropolitan Area Network (MAN) standard based on the distributed queue dual bus (DQDB) access method; this demonstation was part of the switched multimegabit data service (SMDS) offered at 45 Mb/s. Things are moving quickly.

A. The Barriers

Indeed, it is remarkable that ISDN is here at all, given the large number of compelling barriers that it has had to overcome. Primarily, the problem has been that ISDN is a technology developed and desired by the carriers, and not one that was initiated by user demand. As a result, a deadlock persisted that took the following form. First, the carriers were unwilling to deploy a central office ISDN switch until they could estimate the market that would justify the huge expenditures involved. The market could not be estimated until the users judged their likely use of the technology; but the users could not make this judgement until they could be given cost and timing of the ISDN products. To provide this product cost and timing information, the system suppliers needed the chip set cost. But the chip manufacturers were unwilling to tool up until they could see the market that could not develop until the central office switches were in place. This deadlock could only be broken by the carriers who did indeed take the first step and got the process moving.

As we unwind from this deadlock, users are concerned that if they buy now and ISDN is a failure, then they will be left stranded with an obsolete technology whereas if ISDN is successful, then costs will drop due to the usual economies of scale. In both cases, the user is motivated to wait; the user is clearly unclear as to when he should jump on the ISDN bandwagon. Further, the real attraction of ISDN will come when the service is ubiquitous and becomes available in all of the locations in which he is interested; but networking technology expands at a slow rate due largely to the enormous cost of providing broad coverage. We have seen this curse of distributed services many times in the past; for example, it occurred with the introduction of telephones, of Federal Express overnight mail, of public packet switched networks, of FAX, of electronic mail, and more.

The problem is further exacerbated by the fact that not all implementations of ISDN products are interoperable; for example, it is the usual case that ISDN adapters from different manufacturers cannot communicate with each other. The average price of an ISDN adapter for a PC today is $1500, whereas adapters for LAN interconnection of PC's sell for less than $800 (and include a microprocessor as well). The full ISDN standard has not yet been finalized by the CCITT. The fact that there is no equivalent of the Corporation for Open Systems (COS) for ISDN leads to the problem of vendor products that are incompatible. The existence of more than one version of a standard is an oxymoron. And the specter of possible changes in the standard or in the unofficial portions of the standard many well cause today's purchased equipment to become obsolete.

B. The Enablers

In spite of the barriers seen by the carriers, the suppliers, and the users to the introduction and deployment of ISDN, these same groups see significant advantages to ISDN that have been hastening its introduction.

The carriers have passed through a number of years of equal access since divestiture, which has produced a highly competitive marketplace. They have been energized to offer more than just transport and to extend their offerings to central-office based services of various types, most of which are dependent upon the introduction of ISDN. Moreover, the flattening demand of PBX equipment has produced a marketplace in which one vendor's gain is the other vendor's loss (i.e., a zero-sum game). Consequently, a carrier must add value to its offerings to differentiate it and to expand the size of its market; ISDN is the vehicle for this added value. The chip manufacturers have long since recognized that the mass-produced memory chip marketplace has been lost to the Japanese. These manufacturers need other markets, and the ISDN chip market is an attractive one for them.

Major corporate users have seen the cost of their separate voice and data networks rise. These users have begun to recognize that an advanced, integrated corporate network offers them a critical competitive edge as well as lower network costs. The additional function being offered by advanced networks is becoming very attractive to them and their top management is being convinced of these facts. ISDN offers a migration path to achieve these goals. The first customers of the ISDN services have been very large organizations with growing networking needs; the large consumer contact firms (e.g., American Express) are quickly moving in this direction.

The success of ISDN depends critically upon the success of the applications that take advantage of its capabilities. Indeed, it is the identification and development of a rich set of applications that will hasten the growth of ISDN more than any other factor. We have seen this phenomenon at work in a number of other network related systems in the past. Packet switching succeeded in the commercial environment largely because of the electronic mail application that it supported. SNA took hold because of the support it provided for transaction processing. PC LAN's have proliferated because of the need to share peripherals and data.

We have yet to identify the hot new application(s) that will drive ISDN steeply up the demand curve. Some of the applications that have been identified so far include automatic number identification (ANI) as well as the ability to turn off ANI, reduced call setup time from 20 s to less than 3 s, the availability of a single access point for digital services (thus eliminating multiple dedicated access lines), the ability to provide video-based telephony, voice-data applications, desktop ISDN links, etc. So far, none of these have sparked a rush to the ISDN market.

Nevertheless, the carriers are overwhelmingly behind ISDN and they will do all in their power to promote it. It is in their interest to do so. In the long run, it will be in the user's interest as well, for the carriers are the ones who will provide the networking infrastructure that is called for. Today's networks are disorganized, expensive, not integrated, slow, complex, difficult to manage, and unable to interoperate with each other; an international standard interface such as ISDN is badly needed. To their credit, the Europeans have been much more aggressive than the North Americans in implementing ISDN. And if you still doubt that the case for ISDN is justified, consider the fact that the less-developed and under-developed regions of the world are anxious to connect to the world standard network. There is no way that each of them can or should establish their own standard. There absolutely must be an available world standard to which they can attach.

ISDN is a technology that allows those who have not kept pace with the growth in networking technology to catch up immediately.

III. Narrowband ISDN is not Enough

The BRI and PRI ISDN offerings are often collectively referred to as *narrowband* ISDN (NISDN) to distinguish them from BISDN. The data rates associated with NISDN are inadequate for many applications of interest. On the one hand, the BRI providing 64 kb/s channels is not a large improvement over today's modems, which provide data service at 9.6 kb/s and 19.3 kb/s and which are widely available. It is also the case that 64 kb/s is a nonstarter for the data transmission speeds to which today's users have become accustomed (e.g., local area networks running at 10 Mb/s and more). The PRI running at 1.54 Mb/s is a clear improvement over BRI, but is no different in available speed than is the popular T1 offerings in use by the community today (so why abandon T1 and introduce new equipment interfaces for PRI?). Add to that the nasty incompatibilities faced by multinational corporations when they find that PRI in Europe is 2.05 Mb/s rather than 1.54 Mb/s in North America; of course, this problem already exists in today's T1 offerings. The PRI rate is still a significant step away from the bandwidth needs of the data processing community: it takes almost 5 min to move a 50-megabyte file at T1 transmission speeds.

From the viewpoint of data networks, the real excitement of ISDN comes about when one discusses the capabilities of BISDN. 155 Mb/s is a real improvement over today's speeds. The 50-megabyte file can now be moved in 2.5 s! The precursor to BISDN is the growing use of the T3 service (45 Mb/s). Indeed, the huge popularity of T1 and the growing popularity of T3 are setting the stage for the introduction of BISDN at 155 Mb/s and 620 Mb/s.

The need for broadband speeds comes from a number of applications. The existence of today's high bandwidth customer premises networks (i.e., local area networks (LAN's) require long distance broadband to interconnect them; LAN interconnetion using switched broadband data service is a clear and current application. The emerging field of teleradiology in which one transmits medical imagery among hospitals, physicians, and patients requires large bandwidths due to the enormous data files; the typical pair of chest X-rays we all get in a routine medical examination requires as much storage as four volumes of the Encyclopedia Britannica. A similar need comes from the field of telepathology, i.e., the transmission of optical images of biological samples. On-line access to supercomputer output showing real-time rotation of complex molecules in three dimensions in full color can be a real bandwidth hog. File server access to rapid scanning of visual and textual data is another application. The growth of CAD applications will be one source of rapid development and deployment of customized ISDN chips.

Indeed, the first applications of BISDN will be in the commercial and scientific sectors. However, following that, a real drive for broadband will be in the residential sector in order to provide entertainment. For example, CATV cable service passes by 86% of American homes, 55% of homes subscribe to CATV cable services, 30% of homes purchase more than one premium movie channel, 10% buy pay-per-view services, and the average home consumes 7 hours of television per day [4]. HDTV will increase the demand for sevices and will place enormous bandwidth requirements on our communication plant. If the FCC allows CATV services to be offered by the telephone companies, it would be a tremendous pull from the demand side for the installation of broadband capability to the subscriber base. Of course, optical fiber will be the medium providing these large bandwidths, and the economies that support fiber installation are already here. Currently, there are well over a million miles of installed fiber in the U.S. It is now less expensive to install fiber than it is to install copper for large office buildings. Fiber to the curb (FTTC) is becoming competitive for new installations, and fiber to the home (FTTH) is under serious consideration already. The appropriate strategy is to begin the FTTC and FTTH installations now, while NISDN is deploying.

Thus the real payoff in the data networks world for ISDN is the promise of BISDN and all the services and capabilities it will bring.

IV. Current Nets are Inadequate

It is clear that the data networks we inherited from the 1980's are inadequate to handle the applications and capabilities required by the 1990's. Today's packet switched data networks have a number of problems with them: They are high cost, they are low speed, they introduce large switching delays, they have relatively high error rates, the switches require too much intelligence, the switches are electronic, there is too much storage in the network, the protocols are too heavyweight, and too much processing is done in the network.

For example, X.25 packet switching networks are serving a real need as they currently exist. However, they are based essentially on 64-kb/s speeds and use heavyweight protocols (they process up to layer 3 at every hop). As an alternate to X.25 packet switching, frame relay is currently being considered for the interim version of fast packet switching, whereby the LAPD link level protocol will be used to perform switching functions at layer 2 without the layer 3 processing overhead [5], [6].

Tomorrow's broadband networks require new architectures to handle the changing requirements. The move from megabits per second to gigabits per second requires dramatic changes in thinking and in structure. In Table 1 we list some of these contrasts.

Table 1 Packet Network Characteristics: Present Versus Future

	Today	Broadband
Packets/s	Thousands	Millions
Bandwidth	64 kb/s	150 M/-620 Mb/s
Bandwidth allocation	Fixed	Dynamic
Services	Voice, data	Integrated voice, data and image
Switch delay	50–100 ms	10 ms
Propagation delay	Insignificant	Dominant
Error control	Link-to-link	End-to-end
Protocols	Heavyweight	Lightweight
Bottleneck	Link bandwidth	Switch bandwidth

The path from today's data networks to those of tomorrow is being paved right now. T3 offerings at 44.7 Mb/s are beginning to penetrate the private networking marketplace. The synchronous optical network (SONET) standard for optical transmission was agreed upon by the CCITT in 1988 [7] and has promoted BISDN product development. The operations, administration, and maintenance (OA&M) portions of the SONET standard should be completed by the end of 1990 and will only require software updates to implement. SONET has laid out a hierarchy of transmission speeds from 51.8 Mb/s up to 13.27 Gb/s and higher. These enormous speeds are fine for point to point communications (assuming the end points can gobble up gigabits per second), but certainly place some outrageous demands on the internal switches in the network.

These very large communication bandwidths have caused a wealth of research and experimentation to take place in the research laboratories in the advanced area of fast packet switching [8]. Fast packet switching will likely use parallel processing architectures in the switch to handle the millions of packets per second mentioned above. There is a number of competing architectures being proposed for the interconnection networks within these switches and many of them use the Banyan switch in one form or another [9]. The advantage of these architectures is that many packets can be switched simultaneously through the switching fabric using the concurrent processing capability of the parallel processors.

A new multiplexing scheme known as asynchronous transfer mode (ATM) [10] has been adopted for BISDN which uses fixed length packets (called cells) of length 53 bytes (48 bytes of data and 5 bytes of header), has highly simplified protocols (no windowing and no processor-intensive work), incorporates no error detection on the data (only on the header), and implements only layer 1 and basic layer 2 functions in the 7-layer OSI standard. ATM provides connection-oriented virtual circuits, handles continuous and bursty data, eliminates the need for multiple TDM channel rates, provides separate signal and information channels, and is independent of the transmission medium. ATM differs from packet switching in the following ways: ATM has fixed length cells (instead of variable length packets); ATM uses highly simplified protocols (instead of processor-intensive protocols); ATM does not do error correction on the data on a link-by-link basis; and ATM does not do any layer 3 operations.

In addition, the IEEE 802.6 committtee has recently approved a protocol for use in MAN's based on the Distributed Queue Dual Bus (DQDB) [11]. This 802.6 MAN standard is compatible with ATM/BISDN and provides a natural addition to the emerging world of Broadband. The common format shared among this MAN standard, ATM, and BISDN greatly simplifies the internetworking problems of the forthcoming broadband era. Meanwhile, the fiber distributed data interface (FDDI) has met with some success as a 100-Mb/s offering [12].

The carriers are beginning to offer their switched multimegabit data service (SMDS) [13] which will probably be the first manifestation of the 802.6 MAN. SMDS has already been demonstrated at 45 Mb/s and will soon be offered on a tariffed basis. SMDS differs from ISDN in that it is a connectionless data service that includes broadcast and multicast features. ISDN, on the other hand, is an integrated voice and data service offering both circuit-switched and packet switched features.

A near-term problem we foresee for the carriers who are to offer these services is the issue of establishing a tariff that will satisfy the end user in matching his patterns of use in the emerging applications.

In the next five years, we can anticipate that X.25 packet switching will migrate to frame relay, to FDDI and then to the 802.6 DQDB via SMDS, finally bringing us to the ATM/BISDN offerings.

As these brave new broadband capabilities develop, it must be understood that our current networks are ill-suited to provide services using these increased bandwidths. We must re-engineer the architecture of our networks to accommodate these bandwidths, a topic we address in the next section.

V. High Bandwidth Networking

Broadband ISDN is the proposed foundation for wide area networks (WAN's) that are capable of supporting applications needing high speed, low latency, rich functionality, and support of mixed media (i.e., voice, data, image, video, graphics, fax, etc.). The market demand for these advanced applications is clearly growing. Furthermore, the core technologies to provide these services are emerging: high-speed switches are being designed, high-speed fiber access networks are being deployed, the SONET hierarchy has been defined, ATM multiplexing techniques are agreed upon, etc. Indeed, technology is solving most of the performance problems we can foresee (link speeds, processor speeds, and memory sizes are increasing on their own).

As we move into gigabit networks, however, we must take a "clean sheet" approach to many of the systems issues [14], [15]. The critical areas to be considered include switching technology, processor interfaces, protocols, connection-oriented communications, routing, layered architectures, and coexistence with carrier environments. We must be prepared to allow different switch technologies to work in the future broadband networks; these include the BISDN fast packet switching techniques, photonic switches, and wave-length division multiplexing (WLDM). The architecture we select must not depend upon which of these happens to be implemented.

As for switching, tomorrow's networks must be prepared to handle packet, circuit, and hybrid switches. Large packets or groups of packets will have to be switched simultaneously; at gigabit bandwidths, one cannot afford the overhead of switching small blocks independently. Sophisticated dynamic bandwidth reservation algorithms must be developed. Multicast algorithms and capabilities must be developed (fiber is point-to-point, whereas satellite and ground-based radio are broadcast and multicast).

Beyond all of these, the question of the network management system is extremely important. Today's nets are reactive, not proactive. We must introduce proactive diagnosis and service restoral before users sense a problem. We need proactive resource management. Since huge volumes of raw data will be flowing into the management control center, we must use thresholds, filters and alerts, and even expert systems, for early problem detection and resolution. These management functions must operate in a distributed fashion for fault containment, privilege definition, and localization of security failures. Multiple classes of service must be supported. Adaptive protocols and error recovery mechanisms must be developed. Indeed, the management of the emerging internetwork is turning out to be the ultimate challenge in distributed systems.

As we consider these problems, it is clear that the carriers have been facing large network problems for most of this century. They understand management, billing, accountability, security, availability, introduction of new technology on a large scale, etc. However, over the last twenty years, the innovations in data networking have come from the data-processing industry, and not from the carriers. (This in spite of the fact that the data-processing solutions have used the underlying carrier plant to establish their data networks). As we move into the broadband era, it is essential that these two (merged) industries cooperate in providing service to the user community. BISDN holds much promise for advanced networking, and the technological and managerial hurdles that must be overcome are best solved jointly by these two industries.

VI. Conclusions

The concept of ISDN was generated from the carriers. Its early growth was much slower than had been promised due to a number of reasons, key among them being the lack of real user demand for the service. However, in the past two years, the narrowband ISDN (NISDN) penetration has accelerated faster than the skeptics had been predicting.

ISDN is the means by which the less advanced users can quickly catch up to today's technology. However, the real payoff will come with BISDN. The data network services and capacity offered by BISDN are truly exciting and advanced. But we must proceed with NISDN before we can achieve BISDN.

The carriers have an enormous investment in ISDN and they are highly motivated to bring about its success. The carriers are the key to the future networking infrastructure for the U.S. and the rest of the world. The data-processing industry cannot "go it alone" in this endeavor; they must cooperate and encourage the carriers. Both groups must agree on common standards for both private and public networking as this infrastructure grows. ISDN is one important step in this direction. Beyond that, however, it must be recognized that a revolutionary approach must be taken in providing the gigabit/second services about which we are talking. The fundamental architecture of our plant must be overhauled significantly; that overhaul is already well underway.

It is perhaps worthwhile to review some of the economic factors that have, and will, affect the architecture of our communication networks. The cost of moving data across a network consists of two important components; the cost of the channels and the cost of the switches. In the early days of communications, the channel was the expensive component (copper wires strung up on telephone poles) and the switch was a poorly paid human operator. As a result, one could afford to waste switch capacity to save on the expensive communications component. Then, a revolution occurred in communications: microwave radio was introduced and this dramatically dropped the cost of the communications component. At the same time, the switch cost dropped (automatic switches in the form of relays and vacuum tubes appeared), but not as dramatically as the channel. Consequently, a reversal occurred where the switch was now more expensive than the communication channel. Now it was sensible to waste communications capacity in order to save on the switch. Thus circuit switching was introduced. In the 1970's, another reversal occurred when integrated electronics (VLSI) appeared, which dramatically dropped the switch cost relative to the communications cost. Once again, we could afford to waste switching capacity in order to save yet more on the communications costs. Thus packet switching was introduced.

That was the past. Let us now peek into the future. Is there anything out there in the near term that will dramatically drop the cost of the switch? Gallium arsenide components will help, but they do not represent a revolutionary change. On the other hand, warm superconductivity, if it comes, would indeed be a dramatic improvement in switch technology. It would allow the wires to be thinner (and still not generate much heat) thereby allowing smaller dimensions (i.e., reduced latency due to the speed of light) and tighter packing. However, warm superconductivity is not a near-term likelihood. Further, photonic switching would be a revolutionary improvement in switch technology. Here, too, we are talking about a laboratory experiment and not a near-term development. So the answer is "no"; we cannot foresee a dramatic improvement in switch technology near term. But how about a revolution in communications? Is there a technology out there that will dramatically reduce the cost of communications? The answer is a resounding "YES!" Indeed it is already taking place, and it is called fiber optics. As stated above, we have well over a million miles of fiber optics in place in the U.S. alone. We are in the midst of the next reversal, which leads us to a situation where communications are plentiful and the bottleneck has once again become the switch. Our networking architectures are undergoing a massive revamping as we move into this environment. Our 1980's architectures are inadequate for the economics and applications of the 1990's.

In response to this current reversal, we see BISDN services coming along, we see fast packet switching architectures, we see ATM, we see the 802.6 MAN, we see LAN developments, we see FDDI, etc. And, once we get all that wonderful technology in place, is it possible that either warm superconductivity and/or photonic switching will come along so as to cause yet a further reversal and thus another reshuffling of the cards? It seems there will be a need for continual improvement of architectures and systems as new technological developments spawn new possibilities and new applications.

As we begin to move through the 1990's we forsee that broadband ISDN will play an important role in bringing about some of the exciting networking developments. A great deal of research has gone into broadband networking in the last few years. The next few years will see development of products and growth in demand. There is no question but that this technology will provide the basis for a ubiquitous communications infrastructure of enormous capacity.

Let us conclude this paper by listing some of the components that we are likely to see in this time frame:

- Worldwide Data Networks
- Advanced Network Machines
- Optical Fiber Networks
- Gigabit/second Networks
- Megapacket/s Superswitches
- Optical Switches
- Pervasive Local Area Networks
- LAN-MAN-WAN Hierarchy
- Processing Satellites
- Intelligent Network Directories
- Continuous Speech Recognition
- Image Communication Mode
- Digital Signal Compression
- Massively Parallel Systems
- Massively Connected Systems
- Neural Networks
- Pervasive Expert Systems.

It is clear from this list that the convergence of data processing and data communications is virtually complete. Distributed information networks are poised to provide the many services required for the emerging information society. ISDN will serve to hasten access to these information networks, eventually providing a major thrust when BISDN products and services begin to roll out.

References

[1] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications.* New York: Wiley, 1976.

[2] IDC White Paper, "ISDN integrated services digital networks," *Computer World*, CW Publishing, Jan. 22, 1990.

[3] W. Stallings, *ISDN: An Introduction.* New York: Macmillan, 1989.

[4] M. Frame, "Broadband service needs," *IEEE Commun. Mag.*, pp. 59-62, Apr. 1990.

[5] S. J. Lowe, "Plugging into frame relay for speed and flexibility," *Data Commun. Mag.*, pp. 54-62, Apr. 1990.

[6] E. E. Mier, "New signs of life for packet switching," *Data Commun.*, pp. 90-106, Dec. 1989.

[7] R. Ballart and Y. C. Ching, "SONET: Now it's the standard optical network," *IEEE Commun. Mag.*, pp. 8-15, Mar. 1989.

[8] J. S. Turner, "Design of an integrated services packet network," *IEEE J. Select. Areas Commun.*, vol. 4, pp. 1373-1380, Nov. 1986.

[9] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *Proc. 1st Annual Symp. on Computer*

Architecture, G. J. Lipovski and S. A. Szgenda, Eds. Gainesville, FL: ACM, pp. 21-28, 1973.

[10] R. Handel, "Evolution of ISDN towards broadband ISDN," *IEEE Network Mag.*, pp. 7-13, Jan. 1989.

[11] G. C. Kessler, "IEEE 802.6 MAN," *LAN Mag.*, pp. 102-116, Apr. 1990.

[12] K. J. Thurber, "Getting a handle on FDDI," *Data Commun. Mag.*, pp. 28-32, June 21, 1989.

[13] "Generic system requirements in support of switched multi-megabit data service," Bellcore Tech. Advisory, TA-TSY-000772, Issue 3, Oct. 1989.

[14] L. Kleinrock *et al.*, "Toward a national research network," Washington, DC: National Academic Press, 1988.

[15] B. Leiner, Ed., "Critical issues in high bandwidth networking," Research Institute for Advanced Computer Science report, Oct. 1988.

Leonard Kleinrock (Fellow, IEEE) received the B.S. degree in electrical engineering from the City College of New York in 1957 and the M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institute of Technology in 1959 and 1963, respectively.

While at M.I.T., he worked at the Research Laboratory for Electronics, as well as with the computer research group of Lincoln Laboratory in advanced technology. He joined the faculty at UCLA in 1963, where he is now a Professor of computer science. His research interests focus on performance evaluation of high speed networks and parallel and distributed systems. He has had over 160 papers published and is the author of five books—*Communications Nets: Stochastic Message Flow and Delay*, 1964; *Queueing Systems, Volume I: Theory*, 1975; *Queueing Systems, Volume II: Computer Applications*, 1976; *Solutions Manual for Queueing Systems, Volume I*, 1982, and most recently, *Solutions Manual for Queueing Systems, Volume II*, 1986. He is a well-known lecturer in the computer industry. He is the principal investigator for the DARPA Parallel Systems Laboratory contract at UCLA. He was a cofounder of Linkabit Corporation. He is also the founder and CEO of Technology Transfer Institute, a computer/communications seminar and consulting organization located in Santa Monica, CA.

Dr. Kleinrock is a member of the National Academy of Engineering, is a Guggenheim Fellow, and a member of the Computer Science and Technology Board of the National Research Council. He has received numerous best paper and teaching awards, including the ICC 1978 Prize Winning Paper Award, the 1976 Lanchester Prize for outstanding work in Operations Research, and the Communications Society 1975 Leonard G. Abraham Prize Paper Award. In 1982, as well as having been selected to receive the C.C.N.Y. Townsend Harris Medal, he was cowinner of the L. M. Ericsson Prize, presented by His Majesty King Carl Gustaf of Sweden, for his outstanding contribution in packet switching technology. In July of 1986, he received the 12th Marconi International Fellowship Award, presented by His Royal Highness Prince Albert, brother of King Baudoin of Belgium, for his pioneering work in the field of computer networks. In the same year, he received the UCLA Outstanding Teacher Award.